FACULTAD DE INFORMÁTICA
UNIVERSIDAD POLITÉCNICA DE MADRID

MÉTODOS DE RESOLUCIÓN DE PROBLEMAS

Aplicación al diseño de sistemas inteligentes (4ª edición)

Martín Molina

Métodos de resolución de problemas: Aplicación al diseño de sistemas inteligentes

Martín Molina

2006

(4ª Edición)

Autor: MARTÍN MOLINA (MARTÍN MOLINA GONZÁLEZ)

Impreso en España

ISBN: 84-96737-07-1
Depósito Legal: M-43528-2006

Fundación General de la U.P.M.

Facultad de Informática U.P.M. Campus de Montegancedo 28660 Boadilla del Monte – MADRID

Indice

Introducción

Desde que en 1982 Allen Newell publicó su artículo que sentaba las bases de la necesidad un nivel adicional para análisis y descripción del conocimiento de sistemas inteligentes, se han sucedido avances importantes en la forma de concebir y diseñar este tipo de sistemas.

Dos aspectos destacan entre los resultados más interesantes de la investigación en la década de los 90 en el campo de la ingeniería del conocimiento. Por un lado, se ha profundizado en el conocimiento sobre cómo las personas resuelven problemas. Esto se ha logrado por la vía empírica mediante un trabajo científico de observación, experimentación por medios computacionales y generalización, llevado a cabo de la siguiente forma:

1) Observación de la forma de resolución de problemas de un profesional en un determinado campo (diagnóstico médico, diseño mecánico, prospección de yacimientos minerales, etc.).

2) Construcción de un simulador informático que emula a la persona en su actividad profesional, lo que da lugar a un modelo computacional de resolución de un problema concreto.

3) Observación, comparación de sistemas similares y propuesta de un modelo general, no universal, aplicable a una determinada clase de problemas.

Entre los investigadores principales que han desarrollado estas tareas cabe citar como más importantes los trabajos sobre las tareas genéricas de Chandrasekaran, los métodos *role-limiting* de John McDermott y las estructuras de inferencia de John Clancey. Los tres autores, por separado, constatan la existencia de aspectos comunes observados en sistemas que simulan el razonamiento humano cuya identificación y generalización puede servir de base para una mejor comprensión de la forma de resolver problemas. Como resultado de esta línea, actualmente se dispone de un primer conjunto de modelos formales, los llamados *problem-solving methods* (PSM), muy útiles en el campo de la ingeniería informática como plantillas de diseño que guían la construcción de nuevos sistemas. Este trabajo de recopilación de métodos, no obstante, en la fecha actual todavía no se ha dado por terminado, sino que sigue abierto para la inclusión de nuevos métodos, revisión de los existentes y nueva clasificación de los mismos.

Por otro lado, como logro adicional en este campo de investigación, se ha creado un nuevo lenguaje descriptivo como complemento a las formas de representación tradicionales. Dicho lenguaje, que ha sido aceptado por un amplio número de autores, se sitúa en un mayor nivel de abstracción y permite la formulación de arquitecturas más complejas de sistemas basados en el conocimiento. Como autores que más han contribuido en este aspecto se pueden citar: Luc Steels que propuso los denominados componentes de la experiencia como forma de unificación de trabajos previos en este campo, Bob Wielinga, Guus Schreiber y Jost Breuker en el campo de metodologías de ingeniería del conocimiento y el equipo de Mark Musen respecto a herramientas. Dicho lenguaje descriptivo ha supuesto además el planteamiento de una nueva generación de herramientas software de ayuda a los técnicos desarrolladores para construcción de este tipo de sistemas.

El propósito principal del presente texto es servir de base como fuente de información de ambos aspectos recientes del campo de la ingeniería del conocimiento. El texto está dirigido a profesionales y estudiantes del campo de la informática que conocen técnicas básicas tradicionales de representación del conocimiento. La redacción del presente texto se ha orientado en primer lugar para recopilar y uniformizar la descripción de métodos existentes en la literatura de ingeniería de conocimiento. Se ha elegido un conjunto de métodos representativo en este campo que no cubre la totalidad de los métodos existentes pero sí los más conocidos y utilizados en ingeniería del conocimiento, presentando una descripción detallada de carácter tanto teórico como práctico.

El texto describe cada método utilizando una notación común a todos ellos, parcialmente basada en los estándares descriptivos seguidos por las metodologías más extendidas. Para cada método se incluyen algoritmos definidos de forma expresa para el presente texto con ejemplos detallados de operación. Ambos aspectos, la uniformización junto a la presentación detallada de la operación, suponen una novedad interesante respecto al estado actual de la literatura sobre este campo, lo que hace el texto muy adecuado para ser utilizado como libro de consulta como apoyo en la docencia de la asignatura de construcción de sistemas inteligentes.

El texto es utilizado como referencia de parte de la asignatura cuatrimestral de *Ingeniería del Conocimiento* de quinto curso en los estudios de Ingeniería de Informática de la Universidad Politécnica de Madrid. En la redacción del texto se han incluido ejercicios y enunciados de exámenes correspondientes a cursos previos en dicha asignatura.

La organización del texto se ha realizado de forma que, tras una introducción de dos capítulos en donde se presenta el concepto general de los métodos, se presentan cada uno de los métodos de forma secuencial mostrando primero los

correspondientes a los problemas de tipo clasificativo y después los de problemas de tipo constructivo. Cada método se describe inicialmente con una descripción general introductoria. A continuación se muestra la organización del conocimiento en donde se ha puesto especial énfasis dado que es uno de los aspectos clave como guía en la actividad de adquisición del conocimiento. Después, se presenta la inferencia mostrando ejemplos de algoritmos formales orientados a facilitar la comprensión detallada de la estrategia de razonamiento de los métodos. Seguidamente se presentan ejemplos de funcionamiento que ilustran la operación con los métodos. Finalmente se presentan ejercicios propuestos que cubren diversos aspectos de la inferencia y del modelado.

Los dos últimos capítulos tienen como fin describir la utilización de los métodos de resolución de problemas. Para ello se describe, en primer lugar, la forma de combinar métodos para diseñar arquitecturas complejas, en donde se muestran ejemplos reales de sistemas inteligentes que se han construido aplicando dichos métodos. Finalmente, el último capítulo presenta el uso de los métodos dentro de metodologías y herramientas de ingeniería del conocimiento.

En resumen, el texto constituye una documentación sobre los métodos de resolución de problemas más conocidos en el campo de la ingeniería del conocimiento, presentada con un enfoque uniforme y detallado, complementado con ejemplos prácticos.

1 El conocimiento en los sistemas inteligentes

La construcción de sistemas de información se ha basado de forma tradicional en un enfoque de proceso de datos siguiendo diversos paradigmas de organización del software que han evolucionado desde las primeras propuestas de programación estructurada a los más recientes enfoques (orientación a objetos, agentes software, etc.). Por otra parte, en el campo de la inteligencia artificial se han desarrollado técnicas de representación del conocimiento y métodos de resolución de problemas que han permitido plantear la arquitectura basada en el conocimiento. Este tipo de sistemas supone una generalización de los sistemas tradicionales más apropiada para la construcción de aplicaciones en áreas profesionales que requieren simular la forma en que expertos humanos razonan para resolver problemas.

En el presente capítulo se describe el enfoque basado en el conocimiento centrándose en aspectos de representación y análisis del conocimiento con el fin de servir de base a los posteriores capítulos en donde se considera el proceso de construcción de sistemas de una forma más estructurada y con ayuda de métodos generales.

1.1 Planteamiento general

Los *sistemas basados en el conocimiento* o *sistemas inteligentes*[1] pueden entenderse como un tipo de sistemas informáticos que se caracterizan porque:

- simulan la forma natural de resolver problemas observada en las personas,
- encuentran la solución del problema mediante un proceso de búsqueda dirigida por criterios específicos de un dominio.

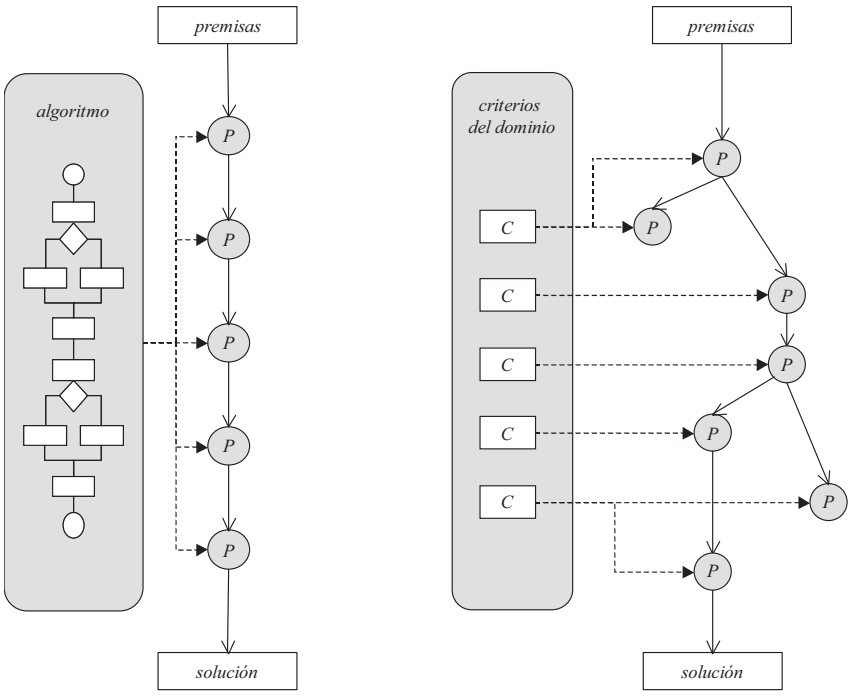


Figura 1.1: Alternativas correspondientes a dos modelos de computación: (1) algorítmico y (2) basado en el conocimiento.

[1] En el presente texto, se utilizarán indistintamente los términos *sistema basado en el conocimiento* y *sistema inteligente*. Recientemente se ha comenzado a utilizar el término *sistema de conocimiento* que resulta muy adecuado si se compara con el de *sistema de información* utilizado en informática convencional.

El esquema básico de computación que se sigue en estos sistemas supone una generalización del esquema tradicional tal como se explica a continuación. La figura 1.1 muestra dos alternativas de computación:

(1) *Modelo algorítmico.* De forma abstracta, la resolución de un problema consiste en realizar un proceso para obtener una solución a partir de unas premisas. Una forma de realizar este proceso es realizar una secuencia lineal de pasos elementales que transforman las premisas en solución. El orden en que se realizan dichos pasos lo indica un algoritmo. Esta opción es la tradicionalmente utilizada en programación bajo el enfoque convencional de tratamiento de información. Es adecuada y resulta eficiente cuando se conoce de forma precisa el algoritmo y éste es estable en el tiempo.

(2) *Basado en el conocimiento*. Sin embargo, en ocasiones hay problemas no bien estructurados en donde el algoritmo no está formalizado (por ejemplo, un problema de arquitectura que busca la mejor distribución de habitaciones en una planta). En esos casos no se conoce con exactitud la forma de llegar a la solución paso a paso aunque se cuenta con criterios que pueden dirigir su búsqueda. En este tipo de problemas, una posible solución es, en vez de un algoritmo global, manejar partes de algoritmos independientes entre sí que expresan cómo realizar pasos locales de transformación de datos. Cada parte de algoritmo se basa en un criterio del dominio o campo en donde se resuelve el problema. La solución se encuentra tras un proceso de búsqueda de aplicación de dichos criterios.

El enfoque algorítmico puede verse como un caso particular del planteamiento más general basado en el conocimiento dado que, en su versión más simple, el conjunto de criterios (partes de algoritmos) tiene un único criterio formulado como algoritmo y que da lugar al proceso de forma lineal puesto que indica en cada momento cuál es el paso siguiente a realizar. En general, el conjunto de criterios del dominio se

basa en un conocimiento parcial del problema y normalmente derivado de la experiencia. Estos criterios se suelen formular haciendo uso de una forma natural de inferencia (reglas, marcos, restricciones, etc.).

Nombre	Descripción
DENDRAL	Determina la estructura molecular de un compuesto orgánico a partir de información proporcionada por un espectrómetro de masas. Utiliza reglas y un descenso jerárquico que reduce la búsqueda de un proceso de generación y prueba. El proyecto se inició en 1965 y durante los años del proyecto, miles de análisis químicos fueron realizados por DENDRAL. Universidad de Stanford. [Lindsay et al., 80]
MACSYMA	Resuelve de forma simbólica problemas de cálculo integral y diferencial. Utiliza reglas de derivación e integración. Se inicia con los trabajos de [Moses, 67] sobre integración simbólica. Instituto Tecnológico de Massachusetts MIT. [Mathlab Group, 77]
MYCIN	Identifica los gérmenes bacterianos que producen una infección y propone una terapia con antibióticos. Utiliza reglas con factores de certeza e inferencia por encadenamiento hacia atrás. El proyecto se inicia en 1972 influenciado por resultados de DENDRAL. Universidad de Stanford. [Buchanan, Shortliffe, 84]
PROSPECTOR	Búsqueda de yacimientos de minerales. Se realizaron diversos modelos para diferentes tipos de minerales. Fue especialmente significativa la detección de un yacimiento rentable de molibdeno. Utiliza reglas con un modelo probabilístico propio. El proyecto se inició en 1976. SRI (Instituto de Investigación de Stanford). [Duda, Reboh, 84]

Figura 1.2: Sistemas basados en el conocimiento pioneros, iniciados en las décadas de los 60 y 70.

Como ejemplo para ilustrar estos enfoques puede considerarse la forma en que las personas resuelven problemas de cálculo diferencial y cálculo integral. Para resolver problemas simbólicos de cálculo diferencial es posible aplicar una secuencia lineal de pasos que permiten encontrar la solución directamente, por lo que puede ser descrito con un enfoque algorítmico. Sin embargo, la resolución de problemas de cálculo integral simbólico no es tan directa. Requiere en ocasiones ensayar diferentes opciones tentativas basándose en diferentes criterios alternativos (integrar por partes, hacer diferentes opciones de cambios de variables, etc.). En este caso es más conveniente aplicar un enfoque basado en el conocimiento.

Sistema	Dominio	Año
ATHENA	Gestión de problemas de hipertensión en cuidados primarios	2002
CEMS	Diagnóstico de enfermedades mentales	1993
DXplain	Ayuda a la decisión clínica	1987
GIDEON	Diagnóstico y tratamiento de enfermedades infecciosas	1994
HELP	HIS basado en el conocimiento	1975
HepatoConsult	Diagnóstico de enfermedades del hígado	1994
Isabel	Enfermedades pediátricas	2002
PAIRS	Ayuda a la decisión de casos clínicos difíciles	2001
QMR	Diagnóstico en medicina interna	1972
RetroGram	Terapias para pacientes de sida	1999
Therapy Edge	Tratamiento del sida	2001
TheraSim CS-HIV	Gestión de enfermos de sida	2002

Figura 1.3: Ejemplos de sistemas de ayuda a la decisión en el campo de la medicina.

Esta solución se ha utilizado con éxito en la construcción de sistemas en dominios profesionales complejos, permitiendo simular de una forma eficaz la forma en que las personas resuelven problemas. La figura 1.2 muestra los primeros sistemas que comenzaron a utilizar esta tecnología con éxito en las décadas de los 60 y 70. Las tablas de las figura 1.3 y 1.4 muestran ejemplos más recientes en el campo de la medicina y en otros campos (diseño mecánico, control de tráfico, experimentación en el espacio, etc.). Sobre aplicaciones existentes recientes pueden consultarse las direcciones web [AAAI, 04] para diferentes tipos de sistemas y [Openclinical, 04] para sistemas en medicina.

Nombre	Descripción
VT	Diseña el sistema mecánico de un ascensor (VT: *Vertical Transport*). Realizado con un método con diversas bases de conocimiento con reglas y restricciones. Utiliza la herramienta general SALT para desarrollo. Sirvió de base para el proyecto SISYPHUS II de comparación de metodologías de ingeniería del conocimiento. Universidad de Carnegie Mellon, Pittsburg Pennsylvania. [Marcus et al, 87]
AIR-CYL	Diseña cilindros que funcionan con aire comprimido para movimiento del pistón y con un resorte en su interior para el movimiento de recuperación. Utiliza el lenguaje DSLP con representación con marcos, reglas y restricciones, que a su vez está programado sobre Lisp. Universidad del estado de Ohio, Columbus [Brown, Chandrasekaran 89]
TRYS/ ESTRADAS	Detecta problemas de tráfico y propone medidas de señalización. Utiliza un enfoque modular con una combinación de representaciones (reglas, patrones, etc). Opera durante años en tiempo real en el Centro Control de Tráfico en Barcelona. El proyecto se inició en 1991. Universidad Politécnica de Madrid. [Molina et al., 98; Molina, Robledo, 01]
(PI)-IN-A-BOX	Asistente para ayudar a astronautas en la realización de experimentos en el espacio con las misiones del *Space Lab* (PI: *Principal Investigator*). Dirige los pasos a realizar en los experimentos y ayuda en la resolución de problemas de ejecución de las tareas. Utiliza reglas de producción (programado sobre CLIPS). Utilizado en el espacio a partir de 1993. Desarrollado por la NASA. [Frainier et al., 94]
RACES	Coordina y planifica la ubicación de aviones de una línea comercial en un aeropuerto. Maneja grupos de restricciones en diferentes fases y de distintos tipos (hard, weak-hard y soft). Planifica 400 vuelos al día en 20 segundos (manualmente se realiza en 5 horas) y permite reasignación. Programado sobre CHIP (Prolog con restricciones). Iniciado en 1995, opera desde 1997 en el aeropuerto de Kimpo (Corea) por la compañía KAL. Inha University (Corea). [Jo et al., 00]
SAIDA	Previsión de avenidas y recomendación de actuaciones. Utiliza una combinación de métodos de detección, predicción y configuración con reglas, marcos, redes bayesianas y cláusulas lógicas con una organización multiagente. El proyecto se inició en 1997. Aplicado en zonas de Levante y Sur de España. Universidad Politécnica de Madrid. [Molina, Blasco, 03].

Figura 1.4: Ejemplos de sistemas basados en el conocimiento.

1.2 Representación del conocimiento

1.2.1 Técnicas de representación simbólica

La representación explícita y formal del conocimiento sobre un problema requiere el uso de técnicas particulares. En el campo de la representación simbólica del conocimiento dentro de la inteligencia artificial se han propuesto diversas formas de representación. Por ejemplo, la figura 1.5 muestra un resumen de algunas de las técnicas más utilizadas.

Técnica	Descripción
reglas	Representan sentencias condicionales SI-ENTONCES con procesos de inferencia mediante encadenamiento hacia delante o hacia atrás.
marcos	Representan estereotipos de situaciones, objetos, conceptos, etc. La inferencia incluye procedimientos que simulan la equiparación y herencia.
restricciones	Representan relaciones de equilibrio/compatibilidad entre variables. La inferencia se basa en la búsqueda sistemática de soluciones que satisfacen las restricciones.
lógica	Utiliza el lenguaje de la lógica como forma de representación. La inferencia se apoya en procedimientos de demostración automática.
redes bayesianas	Red formada por relaciones causa-efecto entre variables cualitativas con medidas de probabilidad condicional para expresar la influencia causa-efecto.
lógica difusa	Representación continua de la semántica de predicados lógicos mediante funciones de posibilidad.

Figura 1.5: Ejemplos de técnicas de representación simbólica del conocimiento

En general, una forma de representación del conocimiento debe satisfacer los siguientes requisitos:

- *Formal.* La representación no debe presentar ambigüedades. Por ejemplo, el lenguaje natural no se considera representación del conocimiento debido a las ambigüedades que presenta.

- *Expresiva.* La representación debe ser suficientemente rica como para capturar los diferentes aspectos que sea necesario distinguir. Por ejemplo, las fórmulas lógicas de cálculo de predicados constituyen una representación más expresiva que la que se maneja cálculo proposicional.

- *Natural.* La representación debe ser suficientemente análoga a formas naturales de expresar conocimiento. En este sentido, las representaciones matemáticas tradicionales y cuantitativas (por ejemplo, las matrices) pueden resultar muy artificiales para emular procesos de razonamiento.

- *Tratable.* La representación se debe poder tratar computacionalmente, es decir, deben existir procedimientos suficientemente eficientes para generar respuestas a través de la manipulación de los elementos de las bases de conocimiento.

Algunas de las más conocidas técnicas que satisfacen los anteriores requisitos son técnicas básicas como las reglas, los marcos, las restricciones, las cláusulas lógicas, y otras más específicas como las redes bayesianas, lógica difusa, etc. Estas formas de representación han sido ampliamente utilizadas en la construcción de sistemas inteligentes con las que, progresivamente, se ha abordado la construcción de sistemas más complejos.

Para considerar de una forma más estructurada el desarrollo de sistemas en problemas con mayor volumen o complejidad de conocimiento se han propuesto técnicas adicionales que complementan las técnicas básicas. Por ejemplo, los *contextos* es una forma de modularización que se basa en realizar una partición en bases separadas de forma que cada contexto puede corresponder por ejemplo a un área de conocimiento relativamente independiente o a una fase del razonamiento. Los contextos mejoran la eficiencia dado que las búsquedas se hacen de forma local y mejoran el mantenimiento de la base pero son limitados en problemas complejos dado que aportan una estructura plana con un único motor de inferencia.

Otra solución es reunir varias formas de representación en una representación múltiple con un único motor de inferencia. Aunque potencia la representación mediante la suma de varias técnicas, es una solución que puede dar lugar a bases de conocimiento heterogéneas de difícil mantenimiento. Este tipo de soluciones

parciales pueden considerarse útiles en la construcción particular de ciertos sistemas cuyas características hagan adecuadas el empleo de alguna de dichas soluciones. Sin embargo, en problemas complejos es necesario ir a enfoque más avanzados que permitan una adecuada modularización y estructuración.

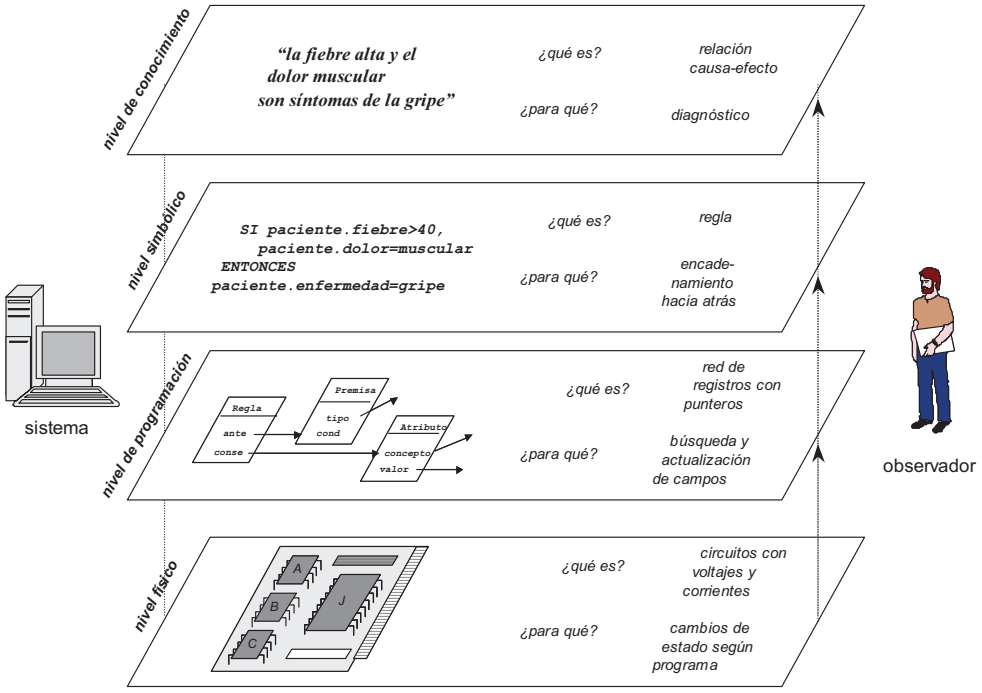


Figura 1.6: Niveles descriptivos de los componentes de un sistema inteligente

Para ello, más recientemente se han propuesto soluciones para análisis del conocimiento y diseño de arquitecturas de sistemas inteligentes que contemplan el manejo de niveles descriptivos adicionales así como recursos de representación que permiten diseñar y reutilizar de una forma eficaz modelos más complejos.

1.2.2. Niveles descriptivos

Con el fin de facilitar la descripción de sistemas inteligentes de una forma más natural y adecuadamente relacionada con los diferentes enfoques computacionales que dan soporte al sistema es conveniente manejar la idea de niveles descriptivos. Un sistema informático puede contemplarse a distintos niveles, según el punto de vista del observador. A nivel más bajo se encuentra el nivel físico en donde el sistema se observa formado por circuitos electrónicos con diferentes estados descritos con voltajes y corrientes. La descripción a este nivel, aunque más cercana a la realidad física que da soporte a la computación, resulta excesivamente artificial para el diseño y programación de sistemas complejos, por lo que raramente se utiliza en esos casos y, en su lugar, se manejan niveles lógicos con entidades descriptivas más naturales y cercanas a la forma en que las personas describen los procedimientos.

Por ejemplo, inmediatamente por encima del nivel físico puede observarse el sistema bajo la perspectiva de la programación convencional (nivel de programación). En este enfoque un sistema se percibe formado por estructuras de datos con registros, campos, etc. además de procedimientos para su manipulación que pueden encapsularse en módulos siguiendo metodologías como el enfoque orientado a objetos. Este es el nivel en el que normalmente se desarrolla el trabajo de implementación haciendo uso de los correspondientes lenguajes de programación.

En la descripción de sistemas inteligentes es útil plantear un nivel superior, al que se puede denominar nivel de representación simbólica, en donde el sistema se contempla formado por bases de conocimiento con representaciones como reglas, marcos, etc. además de procedimientos de inferencia. Este es el nivel en el que se realiza el diseño del sistema inteligente haciendo uso de las técnicas tradicionales de representación del conocimiento del campo de inteligencia artificial.

Sin embargo, como planteó Newell, es interesante considerar además un nivel adicional denominado el nivel de conocimiento. Por ejemplo, tal como muestra la figura 1.6, a nivel de conocimiento una sentencia como *la fiebre alta y el dolor muscular son síntomas de la gripe* se entiende como una relación síntoma-enfermedad y que se puede utilizar para realizar diagnóstico. Sin embargo, a nivel simbólico la misma sentencia se observa como una regla representada con el conveniente lenguaje formal y se utiliza en un procedimiento de búsqueda con encadenamiento hacia atrás. A nivel de programación, la regla se construye mediante la conveniente estructura de datos (por ejemplo, con registros, campos, punteros, etc.) que se maneja mediante un procedimiento de búsqueda y actualización de campos.

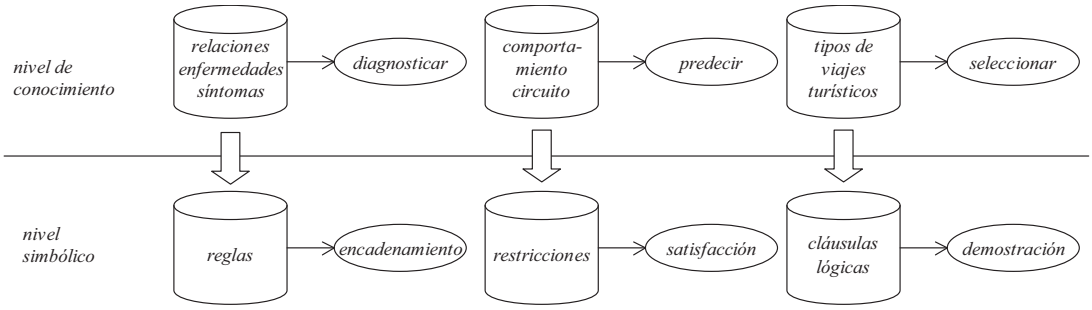


Figura 1.7: Ejemplos de vistas a nivel de conocimiento y a nivel simbólico.

En las actividades de construcción de sistemas inteligentes es importante hacer este tipo de distinciones con el fin de ordenar adecuadamente las actividades de desarrollo manejando el lenguaje descriptivo más adecuado en cada caso. La figura 1.7 muestra ejemplos de bases de conocimiento e inferencias bajo la perspectiva de nivel de conocimiento y su correspondencia a nivel simbólico.

Término	Significado
Conocimiento del dominio	Conjunto de criterios específicos de un determinado campo que sirven de base para resolver un problema. Se separa del conocimiento de inferencia que indica la forma de usar el conocimiento del dominio. El conocimiento del dominio normalmente se representa de forma declarativa.
Conocimiento explícito	Se trata de conocimiento que se puede describir verbalmente mediante análisis introspectivo de los propios procesos de razonamiento. Se contrapone a conocimiento implícito.
Conocimiento superficial	Identifica conocimiento de naturaleza práctica, derivado de la experiencia en resolución de problemas. Se contrapone al conocimiento de naturaleza teórica al que se denomina conocimiento profundo.
Conocimiento de control	Es el conocimiento que determina cuál es el siguiente paso a realizar dentro de un proceso de búsqueda. La elección de adecuados criterios de control es importante desde el punto de vista de la eficiencia de la búsqueda.
Meta-conocimiento	Conocimiento de conocimiento. Derivado de la capacidad humana de ser consciente de su propio saber. Se pueden establecer meta-niveles de forma indefinida.
Conocimiento genérico	Conocimiento que es independiente de un campo específico. Es útil su identificación para plantear su reutilización en distintos dominios.

Figura 1.8: Términos comúnmente aceptados para expresar clases de conocimiento.

1.3. Clases de conocimiento

El estudio de la resolución natural de problemas para construir simuladores que den posibilidad a su automatización requiere llevar a cabo un análisis preciso del conocimiento. Para ello es importante tener en cuenta diversas distinciones y clasificaciones propias del campo de la inteligencia artificial sobre las diferentes clases de conocimiento. En los próximos apartados se describe una terminología comúnmente aceptada dentro de este campo (figura 1.8).

Conocimiento del dominio

Cuando un arquitecto trata de resolver un problema de diseño, aplica ciertas normas urbanísticas y criterios estéticos para encontrar una solución aceptable. En medicina, los médicos manejan relaciones causales entre síntomas y enfermedades para realizar diagnóstico. Cuando se resuelve un problema de asignación de guardias en un centro de atención telefónica se tienen en cuenta ciertas restricciones de equilibrio horario entre los trabajadores. Estos ejemplos muestran diferentes casos de conocimiento especializado en el que se basa la búsqueda de la solución del problema. Dicho conocimiento normalmente está formado por un conjunto de criterios o afirmaciones sobre el campo profesional correspondiente. A este tipo de conocimiento se le denomina *conocimiento del dominio*.

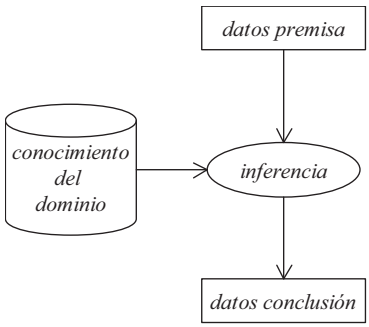


Figura 1.9: Separación entre conocimiento del dominio e inferencia.

Se trata de criterios que habitualmente se expresan sin indicar orden entre ellos y pueden no cubrir todas las opciones posibles. La resolución del problema consiste en realizar una búsqueda (normalmente tentativa) que, a partir de unas premisas dadas, establece una composición o encadenamiento de dichos criterios mediante un proceso deductivo hasta alcanzar unas conclusiones. Este proceso se denomina *inferencia*. Ambos, el dominio y la inferencia, son tipos de conocimiento y su visión de forma separada es la idea central de la arquitectura de los sistemas basados en el conocimiento.

Los criterios que forman el conocimiento del dominio se almacenan en lo que se denomina la base de conocimiento. Dicha base expresa dichos criterios utilizando una *representación declarativa*, es decir, una representación que expresa *qué* se sabe del problema. Por su parte, el conocimiento de inferencia normalmente se codifica en un programa haciendo uso de una *representación procedimental*, es decir, una representación que indica *cómo* se resuelve el problema.

El desdoblamiento dominio-inferencia supone una forma más general de análisis en informática que engloba la visión tradicional del proceso de datos. Bajo este enfoque el desarrollador, en función de cada problema, dará más contenido al dominio o a la inferencia. En caso de que se trate de un proceso convencional de tratamiento de información se contemplará sólo como inferencia. Pero cuando sea necesario hacer uso de criterios explícitos expresados de forma declarativa, tal como se realiza habitualmente en los sistemas inteligentes, será conveniente considerar el conocimiento del dominio.

Conocimiento explícito

Mediante introspección y análisis de los propios procesos de razonamiento las personas podemos describir el conocimiento utilizado en la resolución de ciertos tipos de problemas. Por ejemplo, el médico que establece una conclusión sobre la presencia de una enfermedad puede indicar las relaciones conocidas entre enfermedades y síntomas que ha utilizado en dicha deducción. En esos casos se habla de *conocimiento explícito*, es decir, conocimiento cuya naturaleza y uso en la resolución del problema es posible describir verbalmente. En contraposición a este concepto está el *conocimiento implícito* que se refiere a áreas de saber respecto a habilidades o innatas (por ejemplo saber conducir un vehículo o saber reconocer mediante percepción ciertas imágenes o sonidos) que son difíciles de describir verbalmente.

Es interesante distinguir entre ambos tipos de conocimiento dado que, en el primer caso, será más adecuado manejar formas de *representación simbólica* (reglas, marcos, restricciones, etc.) que pueden identificarse mediante el análisis de las descripciones realizadas por los profesionales en el dominio. Además, el manejo de estructuras simbólicas permite simular la capacidad explicativa de cómo se alcanzan las conclusiones. Por otro lado, para representar áreas de conocimiento sobre habilidades o percepción será más adecuado utilizar *representación conexionista* (por ejemplo, redes neuronales artificiales).

Conocimiento superficial

Mediante el término de *conocimiento superficial* se hace referencia a un conocimiento de naturaleza práctica derivado de la experiencia en la resolución de problemas en un determinado campo. El desarrollo del ejercicio profesional continuado a lo largo de varios años hace que las personas desarrollen unas ciertas habilidades que permiten resolver más rápidamente los problemas sin detenerse en los detalles teóricos que no es necesario hacer explícitos. Dichas habilidades constituyen una forma de conocimiento que puede identificarse y recogerse mediante la conveniente representación. Una gran parte de los primeros sistemas basados en el conocimiento estaban orientados a construir sistemas que representaban conocimiento de este tipo, normalmente en forma de reglas, y que exhibían un comportamiento de resolución de problemas similar al experto que simulaban.

Como contraposición al término anterior puede hablarse del *conocimiento profundo*, que corresponde a un conocimiento más teórico y que, aunque puede requerir procesos de búsqueda más costosos para alcanzar la solución, permite mostrar explicaciones mejor elaboradas. Desde que una persona inicia su vida profesional, tras el periodo de formación, hasta que se considera una persona experimentada, su

conocimiento pasa de ser fundamentalmente de naturaleza profunda a una situación en que dispone además de conocimiento de tipo superficial que mejora la eficiencia en la resolución de problemas. La construcción de sistemas con conocimiento profundo en los años ochenta dio lugar a lo que se denominó entonces sistemas expertos de segunda generación [Steels, 84].

En general hay sistemas sobre los que se resuelven problemas sobre los que se conoce bien su estructura y su comportamiento, por lo que se dispone de modelos suficientemente precisos. Por ejemplo, dentro de este tipo están sistemas mecánicos, eléctricos, etc. sobre los que se pueden resolver problemas de diseño o diagnóstico. Por otro lado, hay sistemas cuya estructura y comportamiento son menos conocidos y en donde se suelen aplicar soluciones de tipo heurístico. Por ejemplo, los dominios de economía y medicina se encuentran habitualmente en esta segunda categoría. En el primer caso se dispone de conocimiento profundo que permiten construir soluciones *basadas en modelos*, es decir, soluciones que utilizan modelos suficientemente precisos de los sistemas sobre los que ser razonan. En el segundo caso se suelen aplicar más soluciones heurísticas que hacen uso de conocimiento superficial.

Conocimiento de control

La inferencia se basa normalmente en la realización de una búsqueda que desarrolla caminos en donde intervienen los elementos de la base de conocimiento y que, en ocasiones, se pueden reconsiderar para ensayar opciones alternativas. En cada paso del proceso de búsqueda es necesario decidir entre varias opciones posibles. Los criterios utilizados para decidir la opción más adecuada forman lo que se denomina *conocimiento de control*. El conocimiento de control influye de una manera directa en la eficiencia del proceso para encontrar la solución dado que unos criterios que

señalen adecuadamente las mejores opciones permitirán alcanzar la solución más rápidamente.

El conocimiento de control puede estar codificado de forma implícita en el programa que realiza la inferencia, representado de forma procedimental. En otros casos puede optarse por utilizar una representación declarativa formando parte de una base de conocimiento adicional. Esto puede visualizarse de una forma recursiva tal como muestra la opción (a) de la figura 1.10, en donde la separación dominio-inferencia se aplica de nuevo en la inferencia para identificar una nueva base de conocimiento, la base de conocimiento de control que contiene los criterios que dirigen la búsqueda sobre la base de conocimiento del dominio. No obstante, esta separación se suele indicar con un esquema como muestra la opción (b) que, aunque no identifica gráficamente cuál es el conocimiento de control (sólo con el nombre), permite mostrar un número indefinido de bases de conocimiento sin que pierda claridad el esquema.

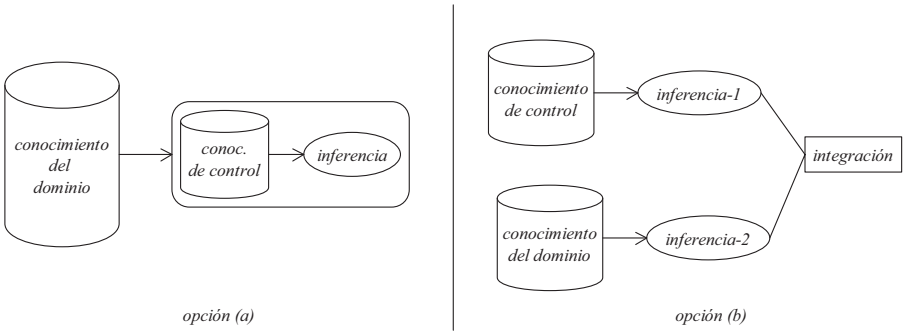


Figura 1.10: Notaciones gráficas para hacer referencia al conocimiento de control.

Metaconocimiento

La capacidad reflexiva de las personas que nos permite ser conscientes de nuestro propio saber nos da la posibilidad de expresar conocimiento sobre lo que ya sabemos. Con el término de *metaconocimiento* se identifica conocimiento sobre el

propio conocimiento. El manejo y simulación de un cierto número de niveles de metaconocimiento puede mostrar un comportamiento más inteligente en un sistema informático, simulando que la máquina es consciente de su propio saber.

Desde el punto de vista de la representación simbólica, el metaconocimiento se caracteriza porque en su vocabulario de conceptos (con el que se expresan las premisas, los hechos intermedios y las conclusiones) se incluyen afirmaciones sobre elementos que están en la base de conocimiento objeto. Por ejemplo, una base de metaconocimiento formulada en reglas puede tener en sus antecedentes y consecuentes identificadores correspondientes a las restricciones de la base de conocimiento objeto que indican las condiciones que hacen que se tengan en cuenta o se supriman ciertas restricciones.

El conocimiento de control puede en ocasiones coincidir con una forma de metaconocimiento. Por ejemplo, en el caso del juego del ajedrez se pueden identificar dos bases de conocimiento: (1) una que contiene los movimientos elementales de las diversas piezas (conocimiento objeto) y (2) otra que establece tipos de aperturas y estrategias generales defensivas o de ataque, expresadas en forma de cadenas de clases de movimientos. Al hacer referencia a los elementos de la primera base de conocimiento, la segunda base se entiende como una base de metaconocimiento aunque, dado que orienta la búsqueda, también es conocimiento de control.

Conocimiento genérico

Al analizar el conocimiento que se maneja en un determinado problema pueden identificarse áreas correspondientes a conocimiento específico y otras que son de carácter más general. El interés en la identificación de conocimiento de tipo general

está principalmente en las posibilidades de reutilización y compartición para reducir el esfuerzo en la construcción de diferentes sistemas.

Normalmente, el conocimiento del dominio suele estar referido a un campo concreto mientras que el conocimiento de inferencia puede describirse de forma más general. También, dentro del conocimiento del dominio pueden encontrarse afirmaciones que son de carácter general y otras de tipo concreto. Por ejemplo, como conocimiento del dominio se pueden tener criterios generales de operación de embalses, independientes de cada embalse junto a otros criterios específicos que expresan formas particulares de operación de cada uno. Desde el punto de vista de análisis se tenderá a identificar contenidos que se puedan establecer de forma general con el fin de ir a diseños más eficientes.

1.4. Análisis del conocimiento

Para llevar a cabo la construcción de un sistema inteligente es necesario identificar y formalizar el conocimiento que se atribuye a la persona experta que sirve de referencia con el fin de emular su aptitud en la resolución de problemas en un determinado dominio. Para identificar dicho conocimiento se realiza una actividad de análisis que da lugar a lo que se denomina *modelo de conocimiento* que identifica las diferentes áreas de saber que dan soporte al proceso de resolución de problemas que se pretende automatizar.

En general, el término *modelo* se define como una abstracción de una realidad observada. Por ejemplo, un modelo de un edificio en forma de maqueta identifica un conjunto de características a pequeña escala abstrayendo otros detalles no necesarios. En el caso del desarrollo de sistemas inteligentes la realidad observada es el conocimiento que se atribuye a la persona experta que resuelve problemas.

Dicha persona, cuando explica su propio razonamiento señala el conocimiento en el que se apoya y que sirve de base para la realización del modelo.

1.4.1. Categorías de conocimiento en el modelo

Básicamente el modelo se expresa con tres categorías de conocimiento: (1) un conjunto de bases de conocimiento que forman la categoría del *dominio*, (2) las bases tienen asociadas inferencias formando la categoría de *inferencia* y (3) las inferencias se combinan mediante métodos para forma la categoría de *tarea*.

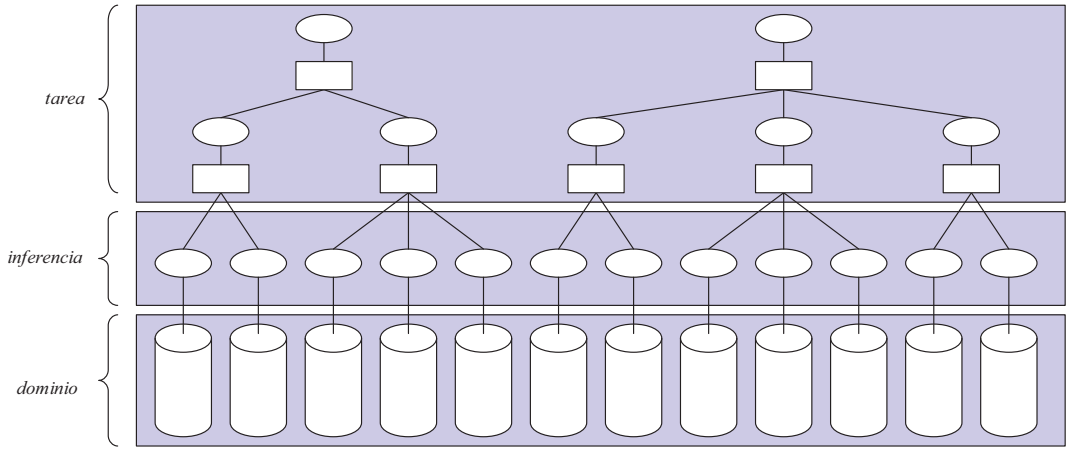


Figura 1.11: Categorías de dominio, inferencia y tarea en un modelo de conocimiento.

Categoría del dominio

La categoría del *dominio* identifica el conocimiento específico del campo en donde aplica el método e incluye el conjunto de criterios particulares de dicho campo que son utilizados por los procedimientos de búsqueda para alcanzar las conclusiones. Normalmente se expresa de forma declarativa. Por ejemplo, corresponde al

conjunto de cuadros clínicos posibles que maneja un médico para hacer diagnóstico o las restricciones de tipo estético que maneja un arquitecto para el diseño de una fachada de un edificio.

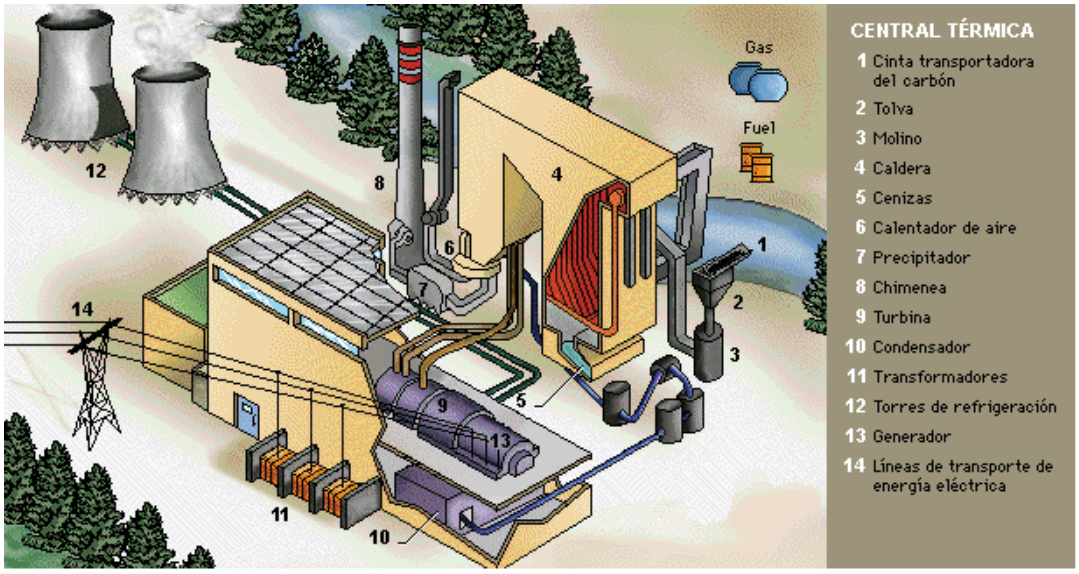


Figura 1.12: Componentes de una planta térmica (www.mestral.org).

El conocimiento del dominio está formado por *bases de conocimiento*. Cada base está formada por: (1) el vocabulario conceptual, que contiene el conjunto de términos que pueden aparecer en la bases (se pueden mostrar en forma de concepto-atributo-valor o predicados) y puede ser común a diferentes bases (2) la estructura de la base (grafo, relaciones causales, etc.), y (3) los elementos de la base que forman cada uno de los criterios que contiene. Para cada base es posible sugerir diferentes formas alternativas de representación simbólica (reglas, marcos, etc.) candidatas a utilizar teniendo en cuenta que, cada realización particular dispondrá de su propia representación.

Para ilustrar más detalladamente la forma en que se describen estas categorías se va a utilizar un ejemplo de problema de diagnóstico de averías en una planta térmica. El objetivo de una planta de este tipo (figura 1.12) es la producción de energía

eléctrica a partir de un determinado combustible, por ejemplo de carbón. La planta dispone de varios componentes para las diferentes fases que se realizan: transporte de carbón, molido del carbón, caldera, turbina, etc. Para la producción óptima de energía los responsables de su operación realizan diferentes observaciones (calidad del molido de carbón, apariencia de la ceniza producida, medidas de temperatura, etc.) lo que les permite identificar fallos de operación mediante un proceso de diagnóstico.

En este caso es posible identificar el siguiente conocimiento del dominio (descrito parcialmente). Se puede tener un vocabulario conceptual descrito en forma de clases e instancias con atributos y valores de la siguiente forma:

```
CONCEPTO depósito ES-UN componente
ATRIBUTOS
  estado {sucio, limpio}
...
CONCEPTO conducto ES-UN componente
ATRIBUTOS
  estado {normal, fuga}
...
CONCEPTO marcador-presión ES-UN marcador
ATRIBUTOS
  estado {baja, normal, alta}
...

CONCEPTO depósito-general ES-UN depósito
CONCEPTO depósito-secundario ES-UN depósito
...
CONCEPTO conducto-principal ES-UN conducto
...
CONCEPTO marcador-presión-principal ES-UN marcador
...
```

La estructura de la base de conocimiento se puede representar mediante un tipo de implicaciones que indican las causas que evocan determinados síntomas observados y otro tipo de implicaciones que describen los efectos que necesariamente producen ciertas averías:

```
CLASE-IMPLICACION evoca
ANTECEDENTE estado(marcador) → CONSECUENTE estado(componente)

CLASE-IMPLICACIÓN manifiesta
ANTECEDENTE estado(componente) → CONSECUENTE estado(marcador)
```

Los elementos de dicha base de conocimiento son las diferentes implicaciones particulares del dominio de la planta tales como las siguientes:

```
IMPLICACIONES evoca:

estado(marcador-presión-principal)=alto
→ estado(depósito-general)=sucio

estado(marcador-presión-principal)=bajo
→ estado(conducto-principal)=fuga
...

IMPLICACIONES manifiesta:

estado(conducto-principal)=fuga
→ estado(marcador-presión-principal)=bajo
...
```

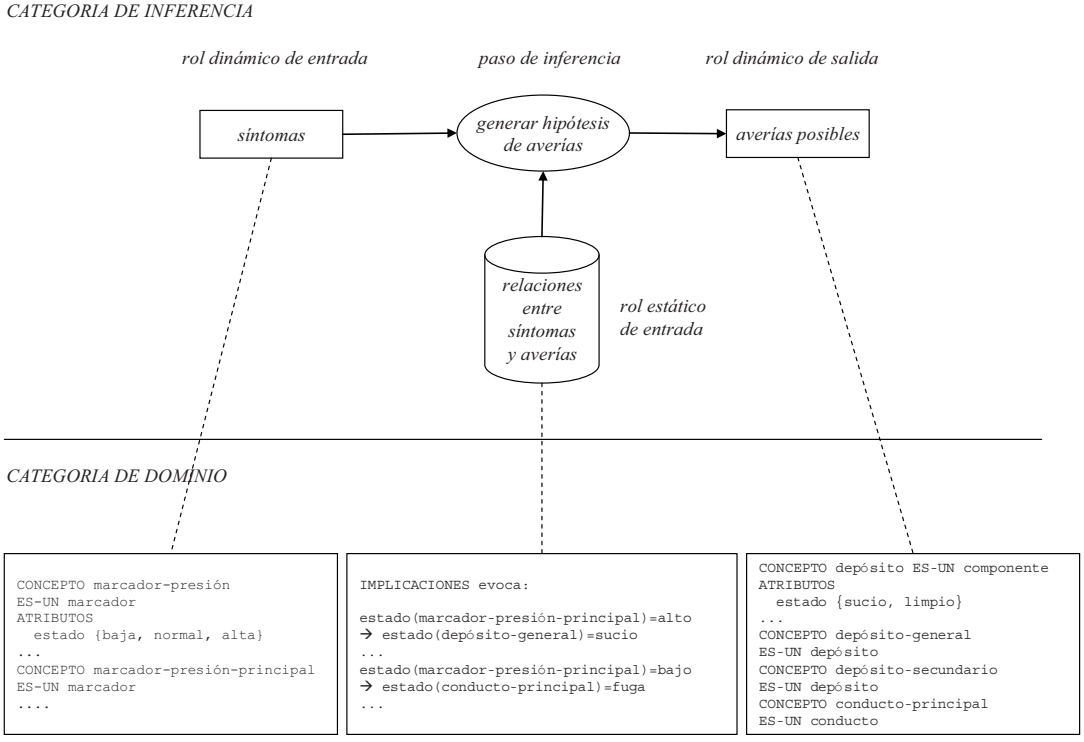


Figura 1.13: Ejemplo de paso de inferencia y su relación con la categoría del dominio.

Categoría de inferencia

Para expresar cómo se utiliza un determinado conocimiento del dominio para resolver un problema se utiliza la categoría de inferencia que está formada por un conjunto de pasos de inferencia. Los *pasos de inferencia* (o sencillamente *inferencias*) son acciones de búsqueda en bases de conocimiento. La inferencia se describe con lo que se denominan *roles de entrada* y *roles de salida*, que indican el papel que juega cierto tipo de información en la inferencia. Los roles de entrada pueden ser *dinámicos* cuando varían en cada problema o *estáticos* si son constantes para diferentes problemas.

Por ejemplo, la figura 1.13 ilustra un caso de inferencia en el dominio de la central térmica junto a la notación gráfica utilizada. El paso de inferencia mostrado se denomina *generar hipótesis de avería* y tiene como fin proponer las posibles averías a partir de diferentes observaciones que muestran un comportamiento anómalo. Tiene como roles de entrada los *síntomas* y las *relaciones entre síntomas de averías*. Como rol de salida tiene las *averías posibles*. En la figura se muestra la relación entre los diferentes roles y el conocimiento de la categoría del dominio.

Obsérvese que esta notación para análisis de conocimiento es una solución adecuada para separar el conocimiento sobre un determinado campo (por ejemplo el concepto conducto o una relación de tipo evoca) en la categoría del dominio, del uso que se le da (el rol) a dicho conocimiento en la resolución del problema en la categoría de inferencia. Esto facilita una descripción en donde un mismo conocimiento del dominio puede tener diferentes roles por ser utilizado por diferentes pasos de inferencia en el proceso global de razonamiento.

Normalmente, en los sistemas basados en el conocimiento un paso de inferencia está asociado a una base de conocimiento sobre la que se realiza un proceso de búsqueda. Cada paso de inferencia, puede implementarse mediante un motor de

inferencia propio de cada representación. Una misma base de conocimiento puede tener una o más inferencias.

En el análisis de sistemas mixtos (basados en el conocimiento y algorítmicos) se pueden incluir inferencias asociadas a bases de conocimiento e inferencias que representan procedimientos algorítmicos y que, por tanto, no tienen base de conocimiento asociada. Esto significa que esta notación es general para ser utilizada en sistemas que incluyen tanto técnicas de representación del conocimiento de inteligencia artificial como técnicas convencionales algorítmicas.

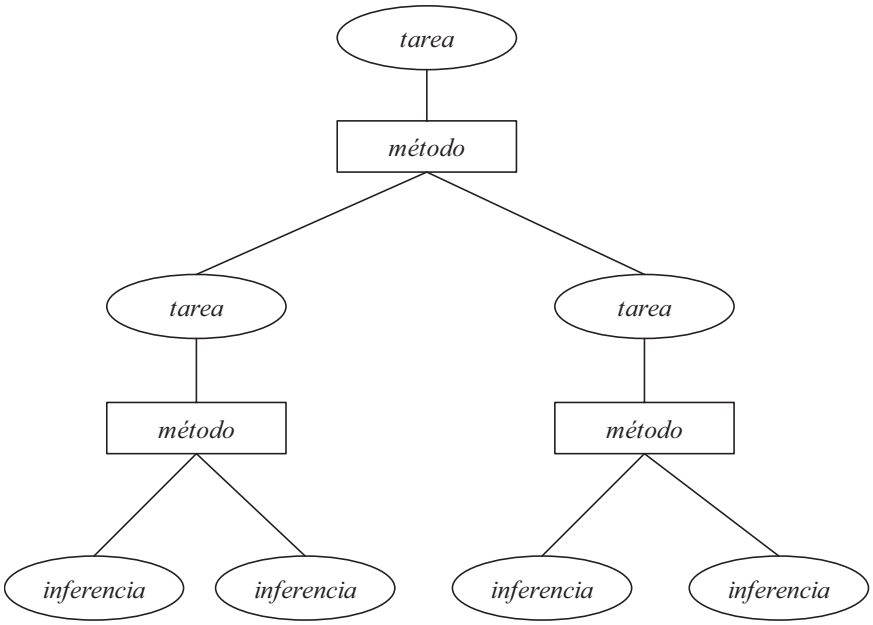


Figura 1.14: Visión gráfica de la descomposición de tareas en subtareas con métodos.

Categoría de tarea

Una *tarea* especifica un tipo de problema a resolver. Ejemplos de tareas son diagnosticar la enfermedad de un paciente, diseñar el sistema mecánico de un ascensor, asignar equipos de trabajo a oficinas, etc. Para describir una tarea, al igual que las inferencias, se indican los *roles de entrada* y los *roles de salida*. Un *método*

expresa cómo llevar a cabo una tarea mediante el uso de subtareas. La elección de un método para una tarea la descompone en tareas más sencillas. Aplicando este proceso de forma recursiva se desarrolla un árbol desde la tarea de más alto nivel hasta tareas elementales que son las inferencias. Gráficamente esta organización se muestra como indica la figura 1.14 en donde las elipses son tareas y los rectángulos son métodos.

Por ejemplo en el caso del problema de diagnóstico en una central térmica, se puede tener la visión de tarea que muestra la figura 1.15. Aquí, el problema general descrito por la tarea de más alto nivel (diagnosticar avería en planta térmica) se descompone en tareas más sencillas, que se identifican en este caso mediante pasos de inferencias.

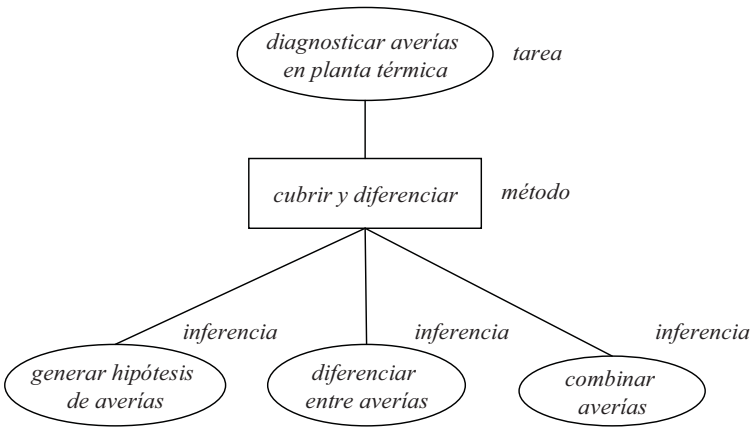


Figura 1.15: Descomposición de la tarea del ejemplo de diagnóstico.

Cada método se describe normalmente haciendo uso de una notación con representación procedimental para expresar cómo alcanza el objetivo expresado por la tarea haciendo uso de tareas más sencillas. En el siguiente capítulo se describe con detalle una notación concreta para formular los métodos.

En conjunto, las tres categorías (dominio, inferencia y tarea) forman una imagen del modelo en forma de jerarquías que parten de tareas de alto nivel y se descomponen hasta llegar a inferencias que operan sobre bases de conocimiento. Dado que la visión de tarea e inferencia, en cierta forma, coincide con la visión de descomposición funcional de análisis en ingeniería del software, esta forma de expresar los modelos permite abordar de una manera uniforme el desarrollo de aplicaciones híbridas con técnicas convencionales y basadas en el conocimiento.

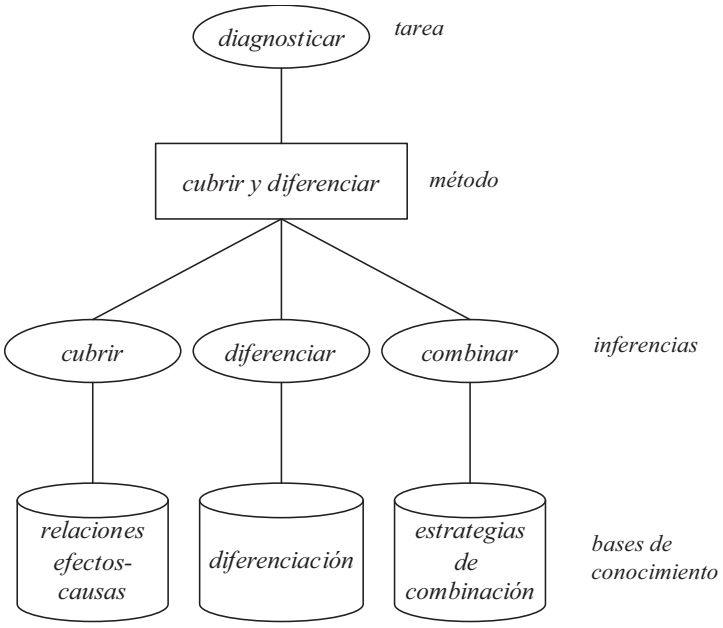


Figura 1.16: Generalización del modelo de conocimiento mediante abstracción.

Esta forma de descripción además permite realizar *abstracción*, lo cual es muy útil para reutilizar modelos de conocimiento en diferentes problemas. Mediante esta abstracción, una estructura se puede describe de forma separada del dominio tal como ilustra la figura 1.16. Dicha figura es una abstracción de la forma de realizar diagnóstico de plantas térmicas y que permite ser reutilizada para otros dominios en donde sea necesario resolver problemas de diagnóstico (por ejemplo, diagnóstico de fallos mecánicos de motores).

1.4.2. Organización distribuida multiagente

Con el fin de facilitar la construcción de modelos más complejos es posible manejar otro tipo de entidades para una organización modular natural. Para ello, por ejemplo, se puede utilizar el concepto de *agente* que aporta un criterio para encapsular en módulos diferentes partes del modelo utilizando un concepto muy intuitivo y natural. Así, un modelo completo al más alto nivel de abstracción se contempla como una colección de agentes que interaccionan para resolver el problema. La inteligencia de cada agente se describe mediante un modelo local de conocimiento haciendo uso de las tres categorías de dominio, inferencia y tarea. En su versión más sencilla un modelo completo tiene un único agente pero los más complejos pueden disponer de una sociedad de diversos agentes que interaccionan de diversas formas.

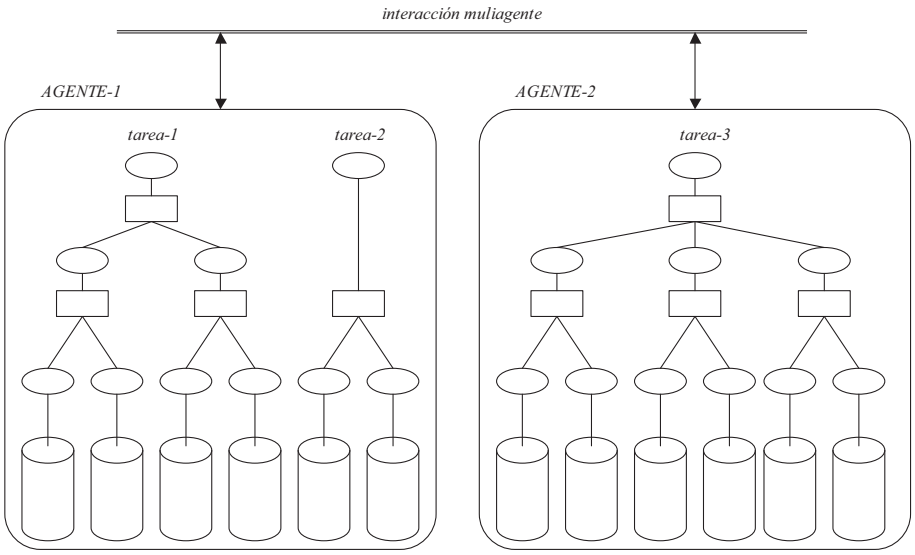


Figura 1.13: Organización distribuida multiagente del modelo de conocimiento.

Uno de los aspectos esenciales en este contexto es la descripción de la interacción entre agentes que puede responder a diversos enfoques que pueden considerar la necesidad de negociar para manejar o acceder a recursos limitados, cooperar para resolver un problema global, etc.

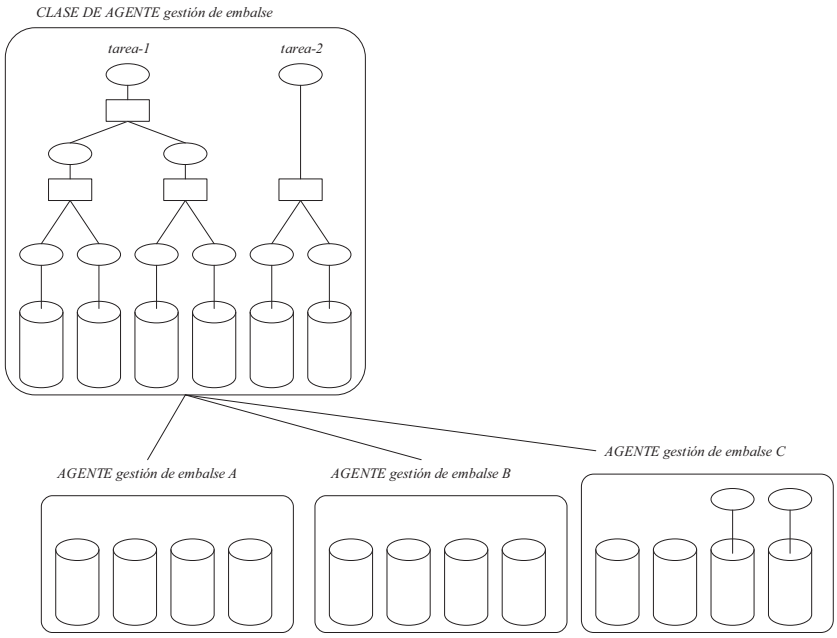


Figura 1.14: Clases de agentes y agentes específicos que comparten conocimiento.

Los agentes se pueden contemplar agrupados en familias o clases de agentes que cuentan con conocimiento y estructura común y, por tanto, que se puede compartir entre diferentes agentes específicos. Así, el conocimiento de gestión de un embalse para un sistema de gestión hidrológica se puede encapsular en una clase de agente indicando cuáles son las tareas, métodos y bases de conocimiento que tiene este tipo de agente. Por otro lado, hay agentes particulares como instancias de dicha clase que heredan la estructura pero que incluyen bases de conocimiento específicas con conocimiento particular y, opcionalmente, variantes a las tareas o métodos tal como muestra la figura 1.14.

La metáfora de organización modular de agente es muy adecuada y natural para identificar unidades que operan de forma autónoma y que interaccionan para resolver problemas de la misma forma que las personas se relacionan en grupos y sociedades, por lo que es muy útil en el desarrollo de sistemas que simulan comportamientos sociales inteligentes en organizaciones.

2 Los métodos de resolución de problemas

La experiencia reciente en la construcción de sistemas inteligentes ha permitido identificar una serie de modelos típicos de sistemas para ciertas clases de problemas como diagnóstico, planificación, configuración, etc. Dado que dichos modelos generales se basan en la simulación de formas de razonamiento observadas en profesionales, se han denominado con el nombre general de *métodos de resolución de problemas*. La investigación en este campo, realizada fundamentalmente en la década de los 90, ha producido la tipificación de un conjunto de métodos que, actualmente, suponen una herramienta muy útil para los desarrolladores de sistemas dado que pueden utilizarse como guía en la construcción de nuevos sistemas.

En el presente capítulo se describen los fundamentos del concepto del método de resolución de problemas. Para ello, se presenta inicialmente una caracterización general de los problemas seguida de la definición y forma de notación de los métodos. Finalmente se muestra un resumen del conjunto de métodos existentes como paso previo a la exposición detallada de los métodos en capítulos posteriores.

2.1. Caracterización de problemas

De forma tradicional en inteligencia artificial un problema se define como la existencia de discrepancias entre un estado actual y un estado objetivo. La resolución recurrente y especializada de ciertos tipos de problemas permite identificarlos en forma de tareas. Una tarea identifica una meta en el razonamiento expresada de forma independiente del dominio con un conjunto de tipos de entradas y salidas.

Bajo este enfoque, una visión general de los tipos de problemas puede expresarse mediante una clasificación de tipos de tareas. En este apartado se muestra una visión que recoge los tipos habituales de problemas que se realizan con sistemas basados en el conocimiento.

2.1.1 Sistemas sobre los que se resuelven problemas

Para describir los problemas de forma abstracta es conveniente contemplar el *sistema* sobre el cual se resuelve cada problema. Por ejemplo, en diagnóstico médico el sistema de referencia es el paciente. En un problema de diseño arquitectónico, el sistema de referencia es el edificio a construir. La figura 2.1 muestra tipos de sistemas que se consideran en el presente texto. En general, un sistema es *simple* si no está formado por componentes y se describe con un conjunto de parámetros en donde cada uno expresa una característica del mismo. Los sistemas se pueden dividir en *dinámicos* si sus características varían en función del tiempo o *estáticos* en caso contrario. Se habla de *sistema de componentes*, cuando está formado por un conjunto de *componentes* que forman una *estructura*. Cada componente puede verse a su vez como un sistema. Un *sistema de recursos* dispone de un conjunto de recursos (por ejemplo, camas hospitalarias o plazas de aparcamiento) para satisfacer unas necesidades.

Para describir los sistemas se utiliza una serie de términos generales comúnmente aceptados (figura 2.2). Las *observaciones* de un sistema corresponden a mediciones y características que pueden ser percibidas directamente por un observador externo. Los *síntomas* o *desviaciones* son observaciones que indican anormalidad o comportamiento no deseado. Los síntomas se explican por la existencia de *causas*. Las causas pueden ser *averías* (componentes averiados) o situaciones que presenta el sistema (por ejemplo infección bacteriana). Los sistemas dinámicos presentan un *comportamiento* descrito como una secuencia *estados* en el tiempo descrito por los valores de sus parámetros. El comportamiento futuro se denomina *pronóstico*.

Tipos de sistemas	**Ejemplos**
sistema simple (objeto)	herramienta de desarrollo informático, tarjeta de crédito
sistema dinámico	población de un país, la bolsa
sistema de recursos	camas hospitalarias, zonas de aparcamiento, servicios de guardias, aulas para docencia, zonas de cultivos
sistema de componentes	distribución de habitaciones en planta de un edificio, composición musical, inversión económica, ruta turística
sistema de componentes dinámicos	motor de gasolina, turbina eléctrica, circuito electrónico, planta térmica

Figura 2.1: Ejemplos de sistemas sobre los que resuelven problemas.

Los sistemas tienen *funciones* para las que están diseñados (por ejemplo, la función de una lámpara es iluminar o la función de un interruptor es dar la posibilidad de cortar la corriente en un determinado punto). Un observador externo puede desear que se fabrique un determinado tipo de sistema. Para expresarlo plantea un conjunto de *requisitos* sobre sus funciones y *preferencias* (coste, calidad, etc.). El sistema puede estar formado por un conjunto de *recursos* (plazas de aparcamiento, camas hospitalarias, etc.) que puede ser necesario gestionar y/o asignar de acuerdo con unas *necesidades* de utilización de recursos expresadas por el agente externo. Sobre un sistema dinámico pueden realizarse *acciones* que pueden modificar el comportamiento. Las acciones se dividen en *acciones externas* si son ajenas al control ejercido por los responsables de la gestión, *acciones de control* si las

realizan los responsables de la gestión y no modifican la estructura del sistema y *acciones de reparación* si modifican la estructura del sistema. Las acciones de control utilizan los denominados los medios que suministra el propio sistema (los *reguladores*) para modificar el comportamiento Un *plan* es conjunto de acciones y tiene como fin alcanzar determinados *objetivos*. Los objetivos se pueden expresar como *acciones generales* que se pretenden realizar, *condiciones* que deben cumplirse, *estados* que se pretenden alcanzar, *síntomas* que se desean eliminar o *causas* que se pretenden eliminar.

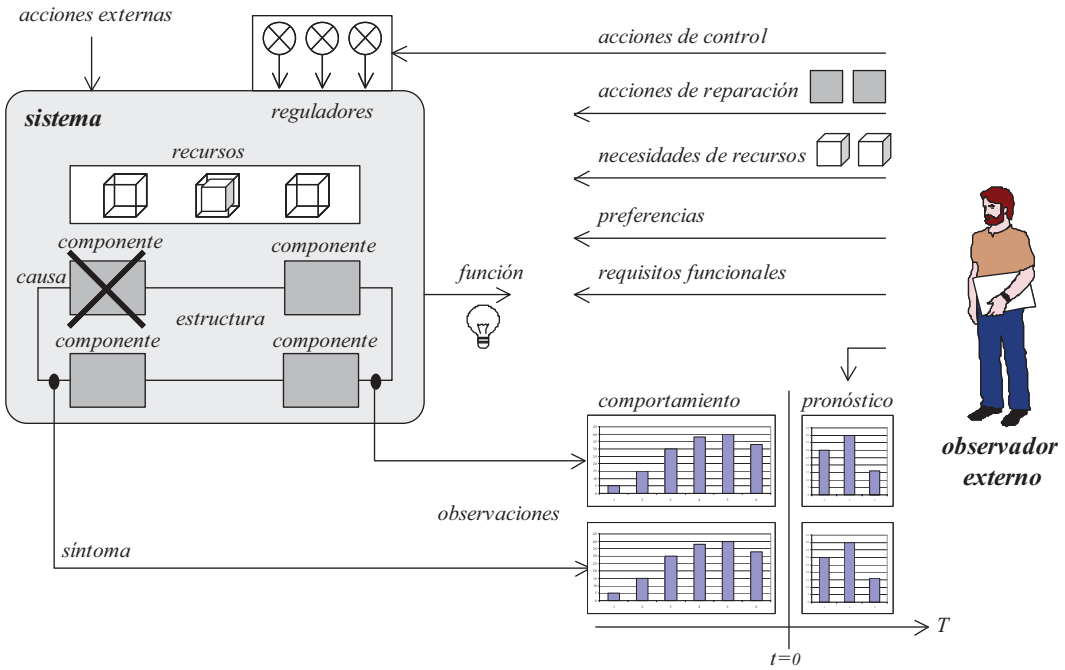


Figura 2.2: Terminología general para describir un sistema sobre el que se resuelve un problema.

Ejemplos

- *Ejemplo 1: Paciente.* En el dominio de medicina, el sistema que puede considerarse como referencia es el *paciente* entendido como un sistema dinámico. El *comportamiento* del paciente corresponde a la evolución reciente de ciertos valores como la temperatura, peso, etc. además del historial clínico. Los *síntomas* son estados del paciente no deseados tales como dolor de cabeza,

temperatura = 40, recuento de leucocitos = 2300. Dichos síntomas se explican mediante enfermedades (*causas*) por ejemplo diabetes o infección bacteriana. El *pronóstico* indica la predicción de la evolución (buen pronóstico, grave, reservado, etc.). Las *acciones de control* corresponden a tratamientos formados por el suministro de medicinas, dietas y terapias. Las *acciones externas* corresponden a acciones relacionadas con el medio en donde reside el paciente (trabajo con estrés, contaminación en ciudad, alimentación alta en calorías, fumador, etc.).

- *Ejemplo 2: Sistema mecánico de ascensor.* El sistema mecánico de un ascensor es un caso de sistema de componentes dinámicos. Para diseñar un sistema de este tipo se parte de un conjunto de *requisitos* sobre las funciones que debe tener el ascensor tales como determinada velocidad de ascenso, número máximo de ocupantes de ascensor, o número de plantas. También se cuenta con *preferencias* tales como el coste máximo del sistema. Los *componentes* del sistema son la cabina, el cable de suspensión, el motor de elevación, contrapeso, amortiguadores, etc. El diseño del sistema se expresa indicando las características técnicas de cada uno de los componentes, por ejemplo, longitud del cable, potencia del motor, modelo de motor, peso del contrapeso, etc. La maquinaria puede presentar ciertas *averías* causados por rotura o desgaste de componentes. Pueden observarse *síntomas* al analizar el comportamiento, por ejemplo, ruidos, desplazamiento defectuoso, consumo excesivo de aceites, etc. La rotura o desgaste excesivo de un componente puede ser debida a *causas externas* tales como defecto en el mantenimiento del sistema o mal uso (peso habitual excesivo, cortes intermitentes de energía eléctrica, humedades e infiltraciones en la zona del motor, etc.). Las *acciones de reparación* se refieren a la sustitución de componentes mediante un *plan* que puede suponer la realización de pasos concretos para desmontar y montar siguiendo ciertos procedimientos establecidos y normas de seguridad.

- *Ejemplo 3. Robot*. Un robot es un caso especial de sistema en donde el observador y el sistema coinciden en el mismo sistema físico, debido a que el robot razona sobre sí mismo para decidir sus propios movimientos. El robot tiene un determinado *comportamiento* dado por los movimientos recientes realizados. El control del robot se basa en la identificación de objetivos que pueden venir dados por la presencia de *desviaciones* con respecto a un *objetivo general* deseado (por ejemplo, la presencia de obstáculos) cuyas *causas* pueden analizarse para encontrar planes como secuencia de acciones que permiten alcanzarlos. Las *acciones de control* son los movimientos que realiza el robot para realizar ciertas *funciones* (cortar el césped, clasificar piezas, etc.). Sobre el robot cabe también plantear problemas de diseño viendo el robot como un sistema de componentes dinámicos que manipula un observador externo con averías, planes de reparación, etc. aunque en este caso no se distingue de cualquier artefacto similar mecánico-electrónico manipulado externamente.

- *Ejemplo 4. Cadena de fabricación*. Una cadena de fabricación es un caso particular de sistema de componentes dinámicos relacionado con otro sistema que es el producto que se persigue fabricar. Por ejemplo, en una cadena de fabricación de automóviles los *síntomas* se observan en el proceso de control de calidad sobre el vehículo final. Así las observaciones se realizan sobre una muestra del producto final sobre el que se pueden detectar desviaciones o síntomas (defectos de pintura, mal ajuste de piezas, etc.). Las *causas* se encuentran en algún punto del proceso de fabricación respecto a mal funcionamiento de ciertos equipos de fabricación, problemas de cantidad o calidad de materia prima, etc. Pueden plantearse decisiones en forma de acciones de control o reparación con respecto a modificar, parar o continuar parcialmente el proceso de fabricación.

- *Ejemplo 5. Red de transporte*. Una red de transporte como es el caso de un conjunto de líneas de autobuses se puede contemplar como un sistema de componentes dinámicos y como un sistema de recursos. En la red se distingue entre la fase de configuración de la red y la fase de gestión o seguimiento de la operación de los autobuses. En la fase de configuración se manejan *recursos* (los autobuses y las líneas) que resuelven unas *necesidades* de transporte establecidas por la demanda de los viajeros. El diseño resultante es la red está formada por un conjunto de *componentes* (las líneas y autobuses) que forman una estructura de recorridos y puntos de intercambio, con unos horarios que tienen en cuenta además de las demandas de viaje, restricciones de los turnos de los trabajadores. Durante la fase de gestión se supervisa la correcta operación de las líneas y se interviene cuando existen problemas. El *comportamiento* global en este caso viene dado por el comportamiento de cada uno de los autobuses que se describe por su localización en cada instante. Los *síntomas* son retrasos en el servicio de los autobuses. Las *causas* de los síntomas pueden ser avería en autobús, corte de calle (manifestación, obras, etc.), retenciones del tráfico, etc. Para resolver las causas se pueden plantear *acciones de control* como incorporar un autobús de reserva (lo que supone resolver un problema de asignación entre vehículos de reserva y necesidades de servicio extra ante la presencia de varios problemas), realizar un itinerario alternativo, etc.

- *Ejemplo 6: Composición musical.* Una composición musical puede contemplarse como un sistema que ha de implantarse sobre otro sistema (interpretarse ante una audiencia). Cuando se compone una melodía normalmente se tienen en mente ciertos objetivos sobre el destino y el efecto en la audiencia que se pueden ver como *requisitos* funcionales (por ejemplo, música disco para baile, música de cine para producir situación de suspense o tristeza). Los *componentes* de una obra musical son frases musicales superpuestas y organizadas sobre los diferentes compases y que constituyen el diseño. Cuando se armoniza una melodía con acordes, cada parte del compás se

asigna un acorde concreto lo que puede verse también como una asignación entre *recursos* (acordes) y *necesidades* (partes de compás con notas a armonizar). Cuando se analiza desde el punto de vista crítico una composición, se pueden identificar *síntomas* o *desviaciones* dadas por disonancias (intervalo de segunda menor) o frases irregulares que rompen un cierto sentido melódico y/o rítmico.

2.1.2 Tipos habituales de tareas

La clasificación de tareas se puede realizar atendiendo a dos criterios principalmente:

a) *Forma de actuación sobre el sistema*. Se distingue entre *problemas analíticos* que son los que extraen conclusiones mediante observación de las características del sistema. Los *problemas sintéticos*, por el contrario, tienen como fin la generación del sistema o partes del mismo mediante síntesis de elementos.

b) *Forma de obtener la solución*. La solución se puede determinar mediante selección dentro de un conjunto prefijado de categorías de solución (*problemas clasificativos*) o mediante construcción de la misma durante el proceso de razonamiento (*problemas constructivos*).

Entre los problemas de tipo analítico se pueden distinguir los siguientes:

- *Clasificar (objeto)*: los datos son *observaciones* sobre características de un determinado objeto (por ejemplo, forma, tamaño, textura, etc.) y el resultado es la *clase* de objeto (categoría) a la que pertenece dicho objeto.

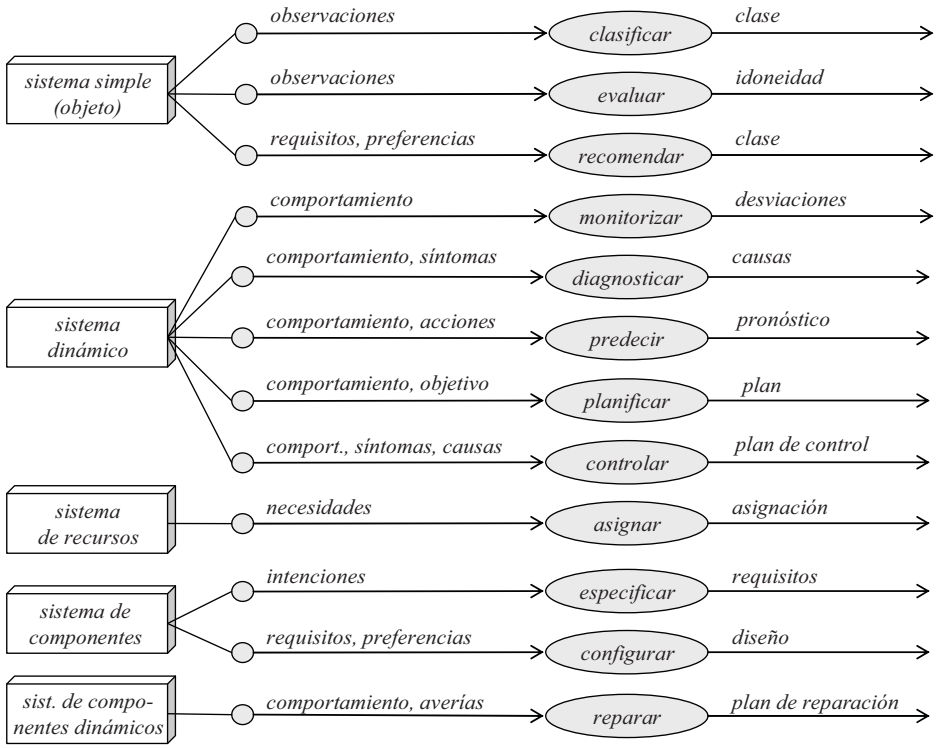
- *Evaluar (objeto):* las entradas son *observaciones* sobre características de un objeto y el resultado es la *idoneidad* de dicho objeto con respecto a un determinado criterio.

- *Monitorizar (sistema dinámico):* las entradas son medidas sobre el *comportamiento* reciente de un sistema y la solución son las *desviaciones* con respecto al comportamiento esperado. Dichas desviaciones (o síntomas) se determinan como resultado de la evaluación de condiciones generales establecidas como objetivos de la gestión de un determinado sistema. Las desviaciones en su versión más simple se determinan mediante el manejo de umbrales para variables aisladas pero pueden requerir la combinación de varias variables (embalse > 90% y lluvia > 20mm/h) o incluso una predicción del comportamiento que indique que en el futuro se pueden superar ciertos valores permitidos.

- *Diagnosticar (sistema dinámico):* como entrada se tiene el *comportamiento* reciente del sistema junto a los *síntomas* que corresponden a situaciones no deseadas y encuentra las *causas* que justifican la presencia de síntomas.

- *Predecir (sistema dinámico)*: las entradas son medidas sobre el *comportamiento* reciente de un sistema además de hipótesis de *acciones* futuras y la solución es el comportamiento futuro (*pronóstico*).

- *Planificar (sistema dinámico)*: las entradas son medidas sobre el *comportamiento* reciente de un sistema y el *objetivo* expresado como un estado o estados a alcanzar; el resultado es el conjunto de *acciones* que permiten alcanzar dicho objetivo. El espacio de soluciones tiene una dimensión de es m^n, en donde n es la longitud media (número de acciones) de los planes posibles y m es el número medio de acciones aplicables en cada estado intermedio.



Término	**Significado**
acción	modifica el comportamiento del sistema (puede ser *externa*, *de control* o *de reparación*)
asignación	asociación entre los recursos del sistema y las necesidades planteadas
avería	componente de un sistema que está averiado y que presenta un mal funcionamiento
clase	familia o categoría a la que pertenece un sistema concreto
causa	causa de un síntoma (estado-causante o componente-averiado) del sistema
componente	subsistema que forma parte de un sistema compuesto
comportamiento	secuencia de estados en el tiempo
diseño	conjunto de componentes detallados y organizados en una estructura
desviación	sinónimo de síntoma
estado	estado que presenta el sistema en forma de valores asociados a magnitudes significativas
estructura	organización de los componentes que forman un sistema
función	la función para la que está diseñado un sistema
idoneidad	decisión para aceptar como válido un determinado sistema
intención	principio general sobre el que se basa la especificación de requisitos
necesidad	necesidad de recursos de un sistema
objetivo	objetivo que se desea alcanzar en forma de estado o condiciones sobre estados
observación	valor obtenido por medición u observación del sistema
plan	conjunto de acciones
preferencia	preferencia sobre cierta características no funcional (coste, estructura, etc.) que debe tener el sistema
pronóstico	comportamiento futuro del sistema
recurso	medio que tiene un sistema para satisfacer cierta necesidad
regulador	medio que permite cambiar el estado del sistema sin cambiar su estructura
requisito	requisito sobre funciones que debe realizar un sistema
síntoma	observación presente o situación futura no deseada que indica anormalidad
sistema	objeto, equipo o instalación artificial, estructura natural o individuo sobre el que se resuelve el problema

Figura 2.3: Tipos habituales de tareas.

- *Controlar (sistema dinámico)*: las entradas son medidas sobre el *comportamiento* reciente de un sistema además de un objetivo expresado como (1) *síntomas* a eliminar, (2) *causas* de síntomas a eliminar; el resultado es el conjunto de *acciones de control* que permiten alcanzar dicho objetivo. Controlar es un caso particular de la tarea de planificar en un sistema dinámico.

Todos estos problemas, excepto los dos últimos, son de tipo clasificativo dado que la búsqueda de la solución se puede realizar dentro de un conjunto prefijado categorías. Los problemas tipo sintético incluyen las siguientes tareas:

- *Recomendar (objeto)*: tiene como dato los *requisitos* funcionales junto a las *preferencias* (coste, calidad, etc.) que se desea que cumpla un objeto y la solución es la clase de objeto que mejor los satisface. Este tipo de problema es clasificativo dado que el espacio de soluciones es el conjunto prefijado de tipos de sistemas a recomendar.

- *Asignar (sistema de recursos):* recibe como dato un conjunto de *necesidades* de recursos del sistema y encuentra una *asignación* entre necesidades y recursos. El problema es de tipo constructivo dado que la asignación se crea cuando se resuelve el problema (el espacio de soluciones tiene una dimensión de $n!$, en donde n es número de recursos o necesidades en cada conjunto, suponiendo igual número).

- *Especificar (sistema de componentes):* los datos de entrada son *intenciones* que expresan los principios generales sobre los que generará como resultado definición de unos *requisitos* funcionales. Las intenciones representan una clase general de requisitos pero definidos de forma imprecisa que debe detallarse mediante interacción con el usuario y mediante cierto conocimiento sobre las posibles características técnicas del sistema a diseñar. La realización de esta

tarea desarrolla un diálogo con el usuario mediante el cual se van deduciendo los requisitos. Es un problema de tipo constructivo dado que no se prefijan todas las combinaciones posibles de requisitos en un espacio de soluciones.

Tarea	**Ejemplos**
CLASIFICAR (objeto)	clasificar una especie animal, clasificar tipo de plantas, clasificar minerales, identificar delitos, identificar formaciones de estrellas
EVALUAR (objeto)	evaluar una operación de tarjeta de crédito, evaluar la presencia de yacimientos minerales, evaluar el cumplimiento de la regulación de exportaciones, evaluar instalación industrial ante seguridad o incendio, evaluar concesión de crédito bancario, evaluar un candidato para un puesto de trabajo, evaluar la afinidad entre dos personas
RECOMENDAR (objeto)	recomendar inversión económica, recomendar tipos de películas de videoclub, recomendar antibiótico más adecuado para una infección bacteriana, recomendar herramienta mecánica para fabricación de componentes, recomendar tipo de cemento, recomendar tipo de madera para construcción de muebles, recomendar tipo de vivienda (agencia inmobiliaria), recomendar carrera universitaria, recomendar deporte a practicar, recomendar cebo en pesca deportiva, recomendar regalos, recomendar libros, recomendar medicinas en farmacias, recomendar automóviles, recomendar mascotas caninas
ASIGNAR(recursos)	asignar colores en un plano, asignar mensajes en paneles de señalización de tráfico, asignar personas a oficinas, asignar plazas de aparcamiento, asignar camas hospitalarias, asignar médicos a guardias hospitalarias, asignar horarios en un colegio, asignar puertas de embarque en un aeropuerto
CONFIGURAR (sistema de componentes)	configurar chalet a la medida de las necesidades de clientes, configurar un equipo informático, configurar un vehículo para participación en rally, configurar vehículo a la medida de un cliente, configurar los extras de un autobús (video, butacas, maletero, cristalería, etc.), configurar el equipo deportivo de montaña, configurar la decoración del hogar, configurar club deportivo (estadio, plantilla, marketing, equipo, etc.), configurar un festejo, configurar una ruta turística, configurar dieta alimenticia, configurar una estructura de andamios para presupuesto
DIAGNOSTICAR (sistema de componentes dinámicos)	diagnóstico de fallos de motor de avión, diagnóstico de fallos en una red de distribución de agua, diagnóstico de fallos en una red telefónica, diagnóstico de fallos en una red eléctrica, diagnóstico de fallos en una red de ordenadores, diagnóstico de fallos en televisores, diagnóstico de fallos en sistema de riego en jardín privado, diagnóstico de fallos en sistema mecánico de automóvil, diagnóstico de fallos de construcción de un edificio,
DIAGNOSTICAR (cadena de fabricación)	diagnóstico de fallos en la construcción de tarjetas de circuitos electrónicos, diagnóstico de fallos en la construcción de botellas de vino, diagnóstico de fallos en el proceso de pintura de automóviles, diagnóstico de fallos en de envasado de alimentos
DIAGNOSTICAR (paciente)	diagnóstico de enfermedades de medicina interna, diagnóstico de enfermedades de las articulaciones, diagnóstico de enfermedades de plantas del hogar, diagnóstico de enfermedades de ganado vacuno, diagnóstico enfermedades infantiles, diagnóstico enfermedades respiratorias, diagnóstico enfermedades del aparato locomotor.
PLANIFICAR (sistema dinámico)	planificar movimientos de cámara en escenarios de realidad virtual,, planificar los pasos de fabricación de componentes de un motor, planificar misiones de ataque, planificar la reforma de una vivienda, planificar la respuesta ante un incendio, planificar rodaje de películas
MONITORIZAR (sistema dinámico)	monitorizar las constantes de un paciente, monitorizar el comportamiento de las inversiones económicas, monitorizar la maquinaria de una turbina generadora de energía eléctrica, identificar problema de tráfico mediante análisis de detectores
PREDECIR (sistema dinámico)	predecir daños producidos por inundación, predecir comportamiento de la bolsa
GESTIONAR (sistema dinámico)	control domótico, gestión de embalses en una red hidrológica, control de una red de comunicación de datos, gestión de una red de autobuses

Figura 2.4: Ejemplos de tareas.

- *Configurar (sistema de componentes):* a partir de unos *requisitos funcionales* y *preferencias* encuentra como resultado un *diseño* que las satisface. El diseño se expresa con un conjunto de componentes y una estructura que los organiza de una determinada forma. El problema es de tipo constructivo y el espacio de soluciones tiene una dimensión de m^n, en donde m es número medio de parámetros que caracterizan todos los componentes y n el número medio de valores de atributos.

- *Reparar (sistema de componentes dinámicos)*: las entradas son medidas sobre el *comportamiento* reciente de un sistema además de unas determinadas *averías* a eliminar expresadas en forma de estados de avería de componentes (o directamente componentes); el resultado es el conjunto de *acciones de reparación* que permiten eliminar las averías. Controlar es un caso particular de la tarea de planificar en un sistema dinámico.

```
evaluar(objeto) = clasificar(idoneidad DE objeto)
        DONDE idoneidad = clase
recomendar(objeto) = clasificar(objeto)
        DONDE requisitos + preferencias = observaciones
monitorizar (sistema dinámico) = clasificar(desviaciones DE sistema dinámico)
        DONDE comportamiento = observaciones, desviaciones = clases
diagnosticar(sistema dinámico) = clasificar (causas DE sistema dinámico)
        DONDE comportamiento + síntomas = observaciones, causas = clases
predecir(sistema dinámico) = clasificar(pronóstico DE sistema dinámico)
        DONDE comportamiento + acciones = observaciones, pronóstico = clase
controlar(sistema dinámico) = planificar(sistema dinámico)
        DONDE NO(síntomas) + NO(causas) = objetivo, control = plan
controlar(sistema dinámico) = configurar(control DE sistema dinámico)
        DONDE NO(síntomas) + NO(causas) = requisitos, control = diseño
controlar(sistema dinámico) = clasificar(control DE sistema dinámico)
        DONDE comportamiento + síntomas + causas = observaciones, control = clase
reparar(sis. comp. dinámicos) = planificar(sis. comp. dinámicos)
        DONDE NO(averías) = objetivo, reparación = plan
reparar(sis. comp. dinámicos) = configurar(reparación DE sis. comp. dinámicos)
        DONDE NO(averías) = requisitos, reparación = diseño
reparar(sis. comp. dinámicos) = clasificar(reparación DE sis. comp. dinámicos)
        DONDE comportamiento + averías = observaciones, reparación = clase
especificar(sistema de componentes) = configurar(sistema de componentes dinámicos)
        DONDE intenciones = requisitos, requisitos = diseño
```

Figura 2.5: Abstracción de tareas mediante transformación de roles. Por ejemplo, la tarea reparar un sistema de componentes dinámicos (novena sentencia) puede verse como la tarea más general de planificar un sistema de componentes dinámicos teniendo en cuenta que el objetivo es que no se tengan averías y que el plan resultante es la reparación.

2.1.3 Abstracción y composición de tareas

Cuando se analiza un problema específico, la misma tarea concreta puede ser vista como diferentes tipos de tareas dependiendo del nivel de abstracción elegido y el sistema de referencia. Por ejemplo, la tarea concreta de recomendar una tarjeta de crédito puede verse como un caso de la tarea general *recomendar(objeto)* o, si se abstrae más, como un caso particular de la tarea general *clasificar(objeto)*.

En general, es deseable disponer de un conjunto mínimo de tareas de forma que el resto de tareas pueda transformarse en alguna de dicho conjunto. Así, se tiende a manejar un nivel de abstracción más general para reutilizar soluciones, lo cual es el objetivo de los métodos de resolución de problemas que se describen más adelante en este capítulo. La figura 2.5 muestra transformaciones habituales de tareas que suponen una abstracción de tareas más específicas a tareas más generales. De acuerdo con dichas transformaciones, el conjunto de tareas mínimo que habitualmente se suele manejar es el siguiente: clasificar, asignar, configurar y planificar. Además, por la existencia de métodos basados en modelos detallados del sistema, también se suele considerar de forma explícita las tareas de predecir y diagnosticar.

```
desarrollar(sistema de componentes) =
       especificar + configurar
gestionar(sistema de componentes dinámicos) =
       monitorizar + [diagnosticar] + [predecir] + controlar
mantener(sistema de componentes dinámicos) =
       monitorizar + diagnosticar + [predecir] + reparar
reconfigurar(sistema de componentes) =
       diagnosticar + configurar
diagnosticar(sistema dinámico) =
       diagnosticar(subsistema-1) + diagnosticar(subsistema-2)
```

Figura 2.6: Combinaciones habituales de tareas. Por ejemplo, la tarea de gestionar un sistema de componentes dinámicos (segunda tarea en la figura) puede verse como la realización de las tareas de monitorizar el sistema de componentes dinámicos seguido de (opcionalmente, por eso se indica con corchetes [ ]) la tarea de diagnosticar, después (también opcionalmente) predecir y, finalmente, la tarea de controlar.

Las clases de tareas se expresan normalmente con un nombre genérico que puede variar en un determinado dominio, lo que implica únicamente una redefinición de términos. Por ejemplo, en el campo de medicina en donde la tarea general *controlar* puede denominarse de forma más específica *prescribir* un determinado tratamiento para actuar sobre una enfermedad.

Por otra parte, es habitual también manejar tareas complejas que son la combinación de otras tareas más sencillas. La figura 2.6 muestra algunos ejemplos de combinaciones de tareas. En este tipo de tareas no existe todavía un acuerdo generalizado entre los diferentes autores sobre los nombres a utilizar aunque sí se suelen utilizar las mismas agrupaciones.

Es interesante aquí hacer notar también que el análisis de tareas puede involucrar más de un sistema de referencia. Por ejemplo para evaluar la idoneidad de un candidato con respecto un puesto de trabajo se pueden manejar dos sistemas, el candidato y la empresa. En ciertas situaciones unos términos de un sistema pueden observarse como términos de otros sistemas relacionados. Por ejemplo, los *requisitos* funcionales de un sistema que se implanta sobre otro (una ruta turística sobre un viajero) pueden ser las *acciones* o *estados* objetivo el sistema sobre el que se implanta (en el ejemplo, un requisito funcional de la ruta turística puede coincidir con el estado objetivo de producir descanso en el viajero). Esto se da, por ejemplo, en problemas que en ocasiones se denominan de planificación aunque realmente se deben tratar como configuración (como es el caso de planificar la ruta turística). Esta forma de planificación, muy presente en el lenguaje usual, centra la atención en la estructura del plan en vez de centrar la atención en el sistema en donde se implanta dicho plan, por lo que es realmente un caso particular de la tarea de configurar un sistema de componentes. Esta diferenciación se trata con detalle más adelante en el capítulo dedicado a planificación.

2.2. Los métodos de resolución de problemas

Con el fin de identificar y tipificar formas habituales de realizar tareas se ha propuesto la idea de método de resolución de problemas. El concepto de los métodos se basa en dos ideas principales: (1) manejo de un nivel adicional de representación (el nivel de conocimiento) y (2) especialización en clases de problemas. La primera idea consiste en situarse en el denominado *nivel de conocimiento* para manejar un plano o nivel de representación en donde el sistema se caracteriza siguiendo medios naturales utilizados en la descripción de las capacidades de las personas. En dicho nivel se indica el contenido de las bases de conocimiento y los roles que desempeñan en el razonamiento de forma independiente de cómo se representa con técnicas simbólicas (reglas, marcos, etc.).

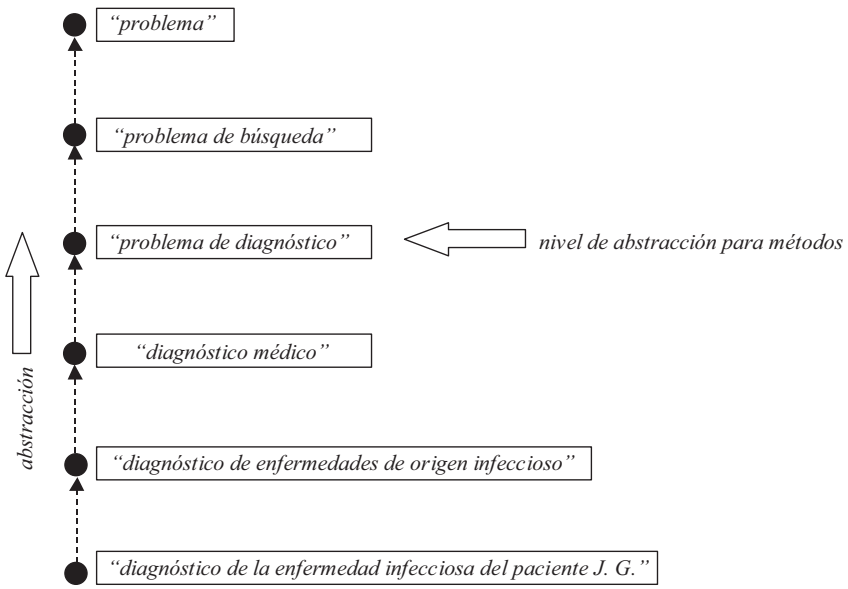


Figura 2.7: Ejemplo de posibilidades de abstracción para describir un problema.

La segunda idea se refiere a la especialización en clases de problemas. La idea es que, en vez de definir métodos universales para la resolución de cualquier tipo de problema, tal como se perseguía en los orígenes de la investigación en inteligencia artificial, se identifican métodos orientados a clases de problemas que resultan más

cercanos al lenguaje de cada problema concreto. Por ejemplo, tal como muestra la figura 2.7 un determinado problema se puede plantear a diversos niveles de abstracción, desde el más concreto, diagnóstico de la enfermedad de un determinado paciente, al más abstracto que se observa como un problema general. La idea de los métodos es situarse a un nivel de abstracción no demasiado concreto para que su potencial de aplicabilidad sea relativamente amplio, pero tampoco demasiado general como para que el uso de ciertas abstracciones haga que el coste de aplicación a un problema concreto sea demasiado elevado.

Siguiendo estas ideas, se ha propuesto el concepto de *método de resolución de problemas* (en la literatura en inglés es frecuente encontrar este término con las siglas PSM por su traducción a *problem-solving method*):

> **Definición 2.1**: Un *método de resolución de problemas* expresa una forma de organizar y utilizar conocimiento del dominio para describir cómo las personas resuelven una clase de problemas.

A diferencia de los métodos originales de resolución de problemas en inteligencia artificial (por ejemplo los métodos de búsqueda heurística tales como búsqueda en espacio de estados, A*, etc.), los métodos de resolución de problemas son de ámbito más restringido y, por tanto, concretan más la organización del conocimiento a seguir. Esta característica ha hecho que se les denominen *métodos fuertes* porque caracterizan con más detalle a dicho conocimiento frente a los métodos generales de inteligencia artificial a los que se les denomina *métodos débiles* porque definen de forma menos precisa, más débil, la organización y uso del conocimiento.

Los métodos de resolución de problemas hacen un uso importante del conocimiento del dominio, lo que aporta un número de supuestos sobre dicho conocimiento que permiten hacer tratables ciertos problemas que de forma general con la algorítmica tradicional pueden ser computacionalmente intratables. Por ejemplo, considérese el

problema arquitectónico de encontrar una distribución de habitaciones en una planta dada que cumpla una serie de requisitos. En este caso, una solución puramente algorítmica podría generar todas las combinaciones espaciales posibles de forma ciega, y aplicar algún tipo de valoración que permita decidir si se ha alcanzado la solución, lo cual sería demasiado costoso computacionalmente debido al gran número de opciones posibles. Como alternativa, un método más razonable haría uso de conocimiento propio del dominio de arquitectura (por ejemplo, normativas de edificación, criterios heurísticos sobre ubicación habitual de espacios, preferencias de tipo estético, etc.) que reduce las opciones posibles permitiendo llegar con un menor esfuerzo a una solución.

Etapa	**Fecha**	**Hechos y propuestas**
Crisis de la representación	principios de los años 80	Críticas a la representación y encuesta SIGART de ACM (1980).
Etapa de propuestas	finales de los años 80	Nivel de conocimiento [Newell, 82], estructuras de inferencia [Clancey, 92], tareas genéricas [Chandrasekaran et al., 92], métodos role-limiting [McDermott, 88].
Etapa de convergencia	principios de los años 90	KADS [Schreiber et al., 93], componentes de la experiencia [Steels, 90], Protege-II [Puerta et al., 93], Sisyphus [IJHCS, 96].
Difusión generalizada	mediados de los años 90	Biblioteca para diagnóstico [Benjamins 93], biblioteca CommonKADS [Breuker, Van de Velde, 94], entornos: Mike [Angele et al. 92], KSM [Cuena, Molina, 97; 00], etc.

Figura 2.8: Evolución histórica de la investigación reciente sobre métodos de resolución de problemas en el campo de inteligencia artificial.

Como indica [Fensel, 00] los problemas abordados por los métodos de resolución de problemas son en general complejos e intratables computacionalmente[1] (excepto ciertos problemas de clasificación que normalmente tienen complejidad polinomial). Para resolver los problemas de forma eficiente los métodos hacen *supuestos* sobre:

- *El dominio*: Expresan supuestos sobre el dominio expresados de forma general (independiente de dominios concretos) sobre condiciones que debe cumplir dicho conocimiento. A su vez aquí se pueden considerar dos tipos de supuestos (1) supuestos sobre la existencia de un determinado tipo de conocimiento, por ejemplo, deben existir relaciones causa-efecto para los síntomas y (2) supuestos sobre propiedades que dicho conocimiento debe tener, por ejemplo, todo efecto tiene que tener al menos una posible causa.

- *La tarea*: Expresan supuestos que restringen el ámbito de aplicabilidad de la tarea general. Por ejemplo, diagnóstico restringido a un solo fallo en vez de múltiples fallos. Otro ejemplo en la búsqueda de una solución es que ésta no tenga que ser necesariamente la solución óptima.

En general, los métodos deben ser correctos y completos. Un método es *correcto* si toda solución propuesta por el método es solución del problema definido por la tarea. Un método es *completo* si se cumple que, si la tarea tiene solución entonces el método debe poder encontrar una solución aunque no necesariamente la misma dado que, en general, puede haber múltiples soluciones.

Los métodos de resolución de problemas identifican aspectos comunes en el razonamiento humano como los siguientes:

[1] Sobre complejidad de los problemas que resuelven los métodos pueden consultarse los trabajos de [Bylander, 91; Bylander et al., 91; Nebel, 96; Goel et al., 87].

- *Simplificaciones de la realidad.* Para razonar sobre una realidad que es compleja, normalmente, las personas realizamos simplificaciones que nos permiten reaccionar de una manera práctica. Dichas simplificaciones vienen dadas en ocasiones en una falta de conocimiento detallado o bien por un resumen de información observada para ganar en eficiencia.

- *Generación y prueba*. La forma de llegar a las soluciones normalmente requiere un proceso de generación de hipótesis inicial y, después, verificación o prueba de dichas hipótesis que puede hacer reconsiderar pasos previos. Este desdoblamiento está basado en que en muchas ocasiones no se puede tener en cuenta todo el conocimiento a la vez para alcanzar la solución, lo que obliga a realizar un pasos de generación de hipótesis con parte del conocimiento, seguido de pasos de verificación mediante otro conocimiento que se encuentra en un segundo plano o que se obtiene mediante la realización de observaciones adicionales. Por ello, los métodos de resolución de problemas normalmente formalizan estrategias de razonamiento que incorporan procedimientos de búsqueda, con fases de tanteo y reconsideración de pasos previos.

- *Jerarquías de niveles de abstracción.* Durante el razonamiento, las personas pueden razonar en niveles altos de abstracción con grandes simplificaciones pero, cuando se requiere, es posible descender a los detalles mediante un proceso de *focalización*, accediendo a un mismo conocimiento pero expresado a más detalle.

- *Descomposición* en *subproblemas*. La forma de abordar la resolución de problemas complejos puede simplificarse mediante descomposición en partes más sencillas. Esta actividad es típica, por ejemplo, en los problemas de planificación.

De forma resumida y, como consecuencia de sus características, los métodos de resolución de problemas tienen las siguientes propiedades:

- *Reutilizables.* Los métodos son reutilizables dado que están orientados a la resolución de clases de problemas (clasificación, diagnóstico, etc.), lo que hace que su rango de aplicabilidad sea relativamente amplio.

- *Más fácilmente utilizables*. Los métodos no se plantean de forma universal con lenguajes demasiado abstractos que los hagan difíciles de aplicar a problemas concretos sino que utilizan lenguajes relativamente cercanos al problema en donde pueden aplicarse, lo que hace que sean más fácilmente aplicables de aplicar que los métodos generales.

- *Naturales*. Los métodos identifican estrategias que han sido observadas en personas, lo que hace que sea muy probable que en la construcción de un nuevo sistema, la forma de razonamiento observada en los expertos coincida con algún método existente.

- *Independientes de la representación*. Dado que los métodos se sitúan a nivel de conocimiento, los métodos no imponen una cierta representación simbólica (reglas, marcos, etc.).

Desde el punto de vista de utilidad en el proceso de construcción de un sistema inteligente, la disponibilidad de los métodos de resolución de problemas tiene las siguientes ventajas:

- *Aporta un lenguaje para análisis y diseño*. Los métodos de resolución de problemas utilizan unos medios descriptivos que aportan un lenguaje para describir a nivel de conocimiento las capacidades de personas especializadas en resolver ciertas clases de problemas. Este lenguaje tiene ciertas

similitudes con los utilizados tradicionalmente en las actividades de análisis de ingeniería del software pero, además, incorpora ciertos elementos específicos para describir las particularidades propias del conocimiento.

- *Sirve de guía en análisis y diseño*. La elección de un método para aplicarlo en la construcción de un sistema sirve de guía en análisis y diseño, en particular en la fase que se denomina adquisición del conocimiento. Cada método define ciertos tipos de conocimiento y su forma de organización, lo que puede ser utilizado como patrón para dirigir el trabajo de adquisición del conocimiento del sistema a desarrollar. Esto ha hecho que el uso de esta técnica se denomine *adquisición del conocimiento basada en modelos*.

- *Permite una mejor simulación de la capacidad reflexiva*. Si se incorpora el método utilizado de forma explícita en la programación con los diversos tipos de conocimiento que se manejan, las explicaciones generadas por el sistema pueden hacer referencia a dichos tipos. Ello puede mejorar la comprensión de las explicaciones de cómo se alcanzan las conclusiones, dado que permite simular una cierta capacidad reflexiva (conciencia sobre el propio conocimiento). Por ejemplo, un sistema de consulta médica para justificar que ha alcanzado la conclusión de que el paciente debe tomar cierta medicina puede mostrar la regla o reglas que ha utilizado en la inferencia, lo cual aporta cierto nivel de explicación. Sin embargo, si además es capaz de indicar los tipos de conocimiento que ha utilizado (conocimiento de interpretación de datos clínicos, conocimiento sobre posibles enfermedades, conocimiento sobre terapias, etc.) hará más comprensible la explicación mostrando cierta conciencia de las clases de conocimiento que posee y, como consecuencia, haciendo más fácil la calibración y puesta a punto de dicho conocimiento.

2.2.1. Tipos de métodos

La figura 2.9 muestra una lista de métodos de resolución de problemas indicando las tareas que realizan. Esta lista incluye los métodos más habituales y con mayor aplicación que se encuentran en la literatura relacionada con el campo de ingeniería del conocimiento.

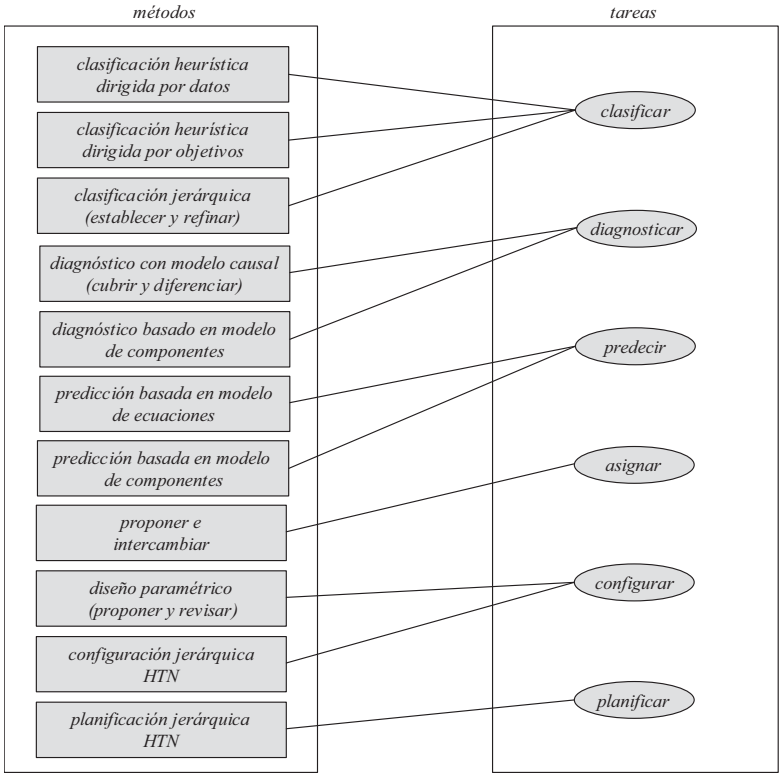


Figura 2.9: Métodos de resolución de problemas (lista parcial).

En este texto se describen de forma detallada la mayor parte de los métodos de dicha tabla. La descripción realizada para cada método debe entenderse como una idealización que sirve como guía, aportando un cierto nivel de abstracción que permite tratar de una manera más sencilla la complejidad del problema. La realización práctica de cada sistema concreto supondrá la necesaria adaptación y extensión de cada método.

2.2.2 Notación para describir métodos

Para describir un método de resolución de problemas se utilizan elementos basados en parte en una generalización y abstracción de los conceptos utilizados en la formulación de modelos de conocimiento. Se manejan los siguientes elementos: (1) la *tarea*, (2) el *conocimiento del dominio* y (3) el *algoritmo del método.* La *tarea* identifica la clase de problema que resuelve el método teniendo en cuenta que, dado que el método está definido de forma general, la tarea no está asociada con un dominio específico. Ejemplos de tareas son diagnosticar, configurar, asignar, etc. sin estar comprometidas con dominios tales como medicina, mecánica de motor, etc.

Tipo/subtipo de conocimiento		**Explicación**	
<tipo-1>	*<subtipo-1>*	*significado*	...
		representación	...
		ejemplos	...
	...	...	...
	<subtipo-k>	*significado*	...
		representación	...
		ejemplos	...
...	...	...	...
<tipo-N>	*<subtipo-j>*	*significado*	...
		representación	...
		ejemplos	...
...	...	...	...

Figura 2.10. Formato de tabla para describir la organización del conocimiento del dominio.

El *conocimiento del dominio* se refiere al conocimiento del campo en donde aplica el método. Es el conjunto de criterios particulares de dicho campo que son

utilizados por los procedimientos de búsqueda para alcanzar las conclusiones y, normalmente, se expresan de forma declarativa. Con el fin de que el método sea aplicable a diferentes dominios específicos, la descripción del conocimiento del dominio se realiza de forma abstracta indicando los diversos tipos de conocimiento que intervienen en el proceso de razonamiento.

Cada tipo de conocimiento se identifica una *base de conocimiento*. El nombre cada base de conocimiento corresponde a un término general que identifica su contenido sin estar comprometido con una representación simbólica ni con un dominio específico. Así por ejemplo una base puede denominarse con el nombre *relaciones causa-efecto*, indicando con ello que contiene relaciones de tipo causal, pero sin indicar en dicho nombre que se representa por ejemplo con reglas y que, además, se aplica al dominio de medicina (dando lugar a relaciones entre enfermedades y síntomas). La misma base puede ser aplicada en la construcción de otro sistema con una representación simbólica distinta (por ejemplo marcos) y en otro dominio, por ejemplo de la mecánica del automóvil.

Rol dinámico	**Significado**	**Representación**	**Ejemplo**
<rol-1>	...	...	..
...	...	...	...
<rol-k>	...	...	..

Figura 2.11: Formato de tabla para describir roles dinámicos.

Sobre el conocimiento del dominio se pueden indicar requisitos en forma de propiedades que deben cumplir los elementos que forman la base de conocimiento para que el método sea aplicable (por ejemplo, exhaustividad de cierto tipo de relaciones, etc.). Para cada base es posible sugerir además diferentes formas alternativas de representación simbólica (reglas, marcos, etc.) candidatas a utilizar teniendo en cuenta que, cada realización particular dispondrá de su propia

representación. La figura 2.10 muestra el formato de la tabla que se utiliza en este texto para describir la organización del conocimiento del dominio. Esta información es uno de los aspectos esenciales de la descripción del método dado que, como ya se ha indicado, puede servir como guía en la actividad de adquisición de conocimiento.

El *algoritmo* del método expresa la estrategia de utilización del conocimiento del dominio para realizar la tarea. Para describir el *algoritmo* del método, se manejan los *pasos de inferencia* (o *inferencias* simplemente) y se describen con los siguientes elementos:

```
INFERENCIA <nombre-inferencia>
        DATOS:                  <roles dinámicos de entrada>
        BASES DE CONOCIMIENTO: <bases de conocimiento>
        RESULTADOS:             <roles dinámicos de salida>
```

Con el fin de mostrar ejemplos de algoritmos detallados de métodos en el presente texto los roles dinámicos se describen indicando su posible representación en tablas cuyo formato muestra la figura 2.11. Dicha representación se basa en el empleo de las siguientes estructuras de datos:

- terna *atributo(concepto)=valor*, por ejemplo `fiebre(paciente)=alta`,
- par *atributo(concepto)*, por ejemplo `fiebre(paciente)`,
- par *atributo=valor* (o *parámetro=valor*), por ejemplo `fiebre=alta`,
- tupla de *n* elementos, por ejemplo `<a,b,c>`,
- valor de `{VERDADERO, FALSO, DESCONOCIDO}` o `{ÉXITO, FRACASO}`,
- estructura, como nombre seguido de argumentos, por ejemplo `P(a,b,c)`,
- conjunto, por ejemplo `{a,b,c}`.

Para referirse a un paso de inferencia dentro de un algoritmo, en el presente texto se manejará la siguiente una notación (basada parcialmente en la notación de la metodología CommonKADS [Schreiber et al., 00]):

```
inferencia(dato-1, ..., dato-N -> result-1, ..., result-M)
```

Por ejemplo:

```
cubrir(síntomas, observaciones -> causas)
```

```
METODO planificación-jerárquica-HTN
DATOS: estado, acción
RESULTADOS: estado, plan, éxito

ALGORITMO
1. guía = {acción}
2. planificar(estado, guía, {} -> estado, plan, éxito)

PROCEDIMIENTO planificar
DATOS: estado, guía, plan
RESULTADOS: estado, plan, éxito
1. IF guía = ϕ THEN éxito = VERDADERO
2. ELSE
3.   GET-FIRST(acción, guía)
4.   IF TIPO(acción) = concreta THEN
5.     aplicar(estado, acción -> estado)
6.     planificar(estado, guía, plan U {acción} -> estado, plan, éxito)
7.   ELSE
8.     seleccionar(acción, estado -> estrategias)
9.     éxito = FALSO
10.    WHILE (estrategias no ϕ) AND no(éxito)
11.      GET (estrategia, estrategias)
12.      refinar(estrategia -> subplanes)
13.      WHILE (subplanes no ϕ) AND no(éxito)
14.        GET (subplan, subplanes)
15.        planificar(estado, subplan U guía, plan -> estado, plan, éxito)
```

Figura 2.12: Ejemplo de descripción del algoritmo de un método.

Notación	Significado
ϕ	Conjunto vacío
<x> := <y>	El valor de <y> se asigna a <x>
<x> = <y>	Condición de igualdad entre <x> e <y>
<c1> SUBSET-OF <c2>	Condición que expresa que <c1> es subconjunto de <c2>
(<a>) AND (<b>)	Conjunción de <a> y <b>
(<a>) OR (<b>)	Disyunción de <a> o <b>
NOT(<a>)	Negación de <a>
{<e>}	Conjunto formado por el único elemento <e>
<c1> $\cup$ <c2>	Unión de conjuntos
<c1> $\cap$ <c2>	Intersección de conjuntos
|<c>|	Número de elementos del conjunto <c> (cardinalidad)
GET(<e>,<c>)	Operación que extrae un elemento <e> del conjunto <c> (queda sin definir cómo se elige <e>)
GET-FIRST(<e>,<c>)	Operación que extrae el primer elemento <e> del conjunto ordenado <c>
GET-FIRST-TYPE(<e>,<c>,<t>)	Operación que extrae el primer elemento <e> del conjunto ordenado <c> que es de tipo <t>
APPEND(<c1>,<c2>)	Función que devuelve el resultado de concatenar el conjunto ordenado <c1> con el conjunto ordenado <c2>
SUBSTITUTE(<c1>,<e>,<c2>)	Función que devuelve el conjunto ordenado resultante de sustituir en el conjunto <c1> el elemento <e> por los elementos del conjunto <c2>. Por ejemplo el resultado de SUBSTITUTE({a,b,c},b,{d,e}) es {a,d,e,c}.
IF <cond> THEN ... ELSE ...	Estructura de control IF-THEN-ELSE
REPEAT ... UNTIL <condición>	Estructura de control REPEAT-UNTIL
WHILE <condición> ...	Estructura de control WHILE
FOR-EACH <e> IN <c> DO ...	Estructura de control de repetición de proceso para cada elemento <e> del conjunto <c>

Figura 2.13: Notación utilizada en las sentencias para formular algoritmos.

El algoritmo del método se describe con una notación en pseudocódigo (con los habituales mecanismos de control utilizados en programación de *repeat*, *if-then*, etc.) haciendo las correspondientes llamadas a los pasos de inferencia previamente definidos. La figura 2.12. muestra un ejemplo completo sobre cómo se describe un algoritmo, para el caso del método de planificación jerárquica. Al comienzo se

indican los datos y los resultados. A continuación se describe el algoritmo que, opcionalmente, como en el caso de la figura, puede incluir procedimientos, lo cual es habitual para describir procesos recursivos. La notación para llamar a procedimientos es la misma que se utiliza para pasos de inferencia.

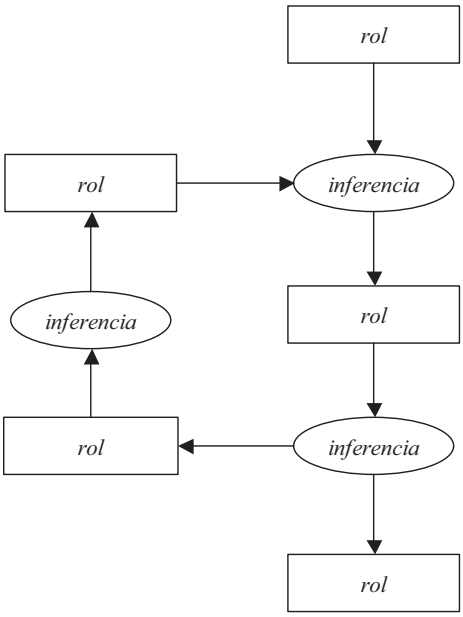


Figura 2.14: Diagrama denominado *estructura de inferencia* que se utiliza para mostrar la conexión entre inferencias y roles dinámicos.

En el pseudo-código se indica de forma destacada con letras en negrita las llamadas a los pasos de inferencia del método. La altura a la que comienza cada línea indica la pertenencia a estructuras de control de tipo *repeat, if-then*, etc. En las líneas del código puede invocarse a operaciones sencillas sobre conjuntos haciendo uso del formato de sentencias que recopila la figura 2.13. En el pseudo-código las frases en cursiva indican sentencias en lenguaje natural que describen operaciones elementales que no presentan ambigüedad.

El algoritmo se puede acompañar de un diagrama denominado *estructura de inferencia* (figura 2.14) que presenta ciertas similitudes con un diagrama de flujo de

datos, en donde mediante elipses se identifican los pasos de inferencia y mediante rectángulos se identifican los roles dinámicos. Las flechas expresan cómo se encadenan los resultados de unos pasos de inferencia como datos de otros pasos de inferencia. Opcionalmente un rol dinámico puede estar enlazado directamente con otro cuando uno es consecuencia casi inmediata del otro (por ejemplo, cuando un rol es un elemento que se selecciona de otro rol que es un conjunto).

Para dar una imagen global de la estructura del método que muestre la tarea, los pasos de inferencia y las bases de conocimiento se suele manejar un gráfico como el que muestra la figura 2.15. Esta imagen es adecuada para dar una visión de conjunto del método y además es útil, como se verá en capítulos posteriores, para mostrar cómo se componen varios métodos para dar lugar a una arquitectura más compleja.

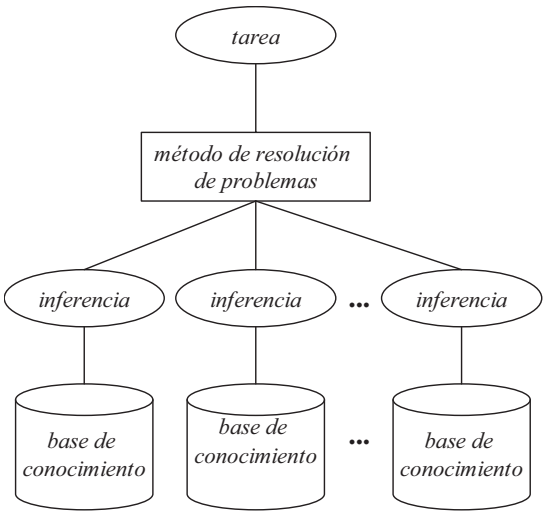


Figura 2.15: Gráfico utilizado para mostrar la estructura global de un método.

En resumen, la notación de los métodos debe entenderse como una estructura abstracta, vista como una forma de plantilla o esqueleto, que servirá de base para una realización concreta de un sistema final, mediante la conveniente particularización de sus elementos en el dominio de aplicación.

2.3. Ejercicios

EJERCICIO 2.1. Con el fin de realizar un ejercicio de análisis de conocimiento e identificación de tareas, considerar los siguientes dominios:

- Operación de una central hidroeléctrica
- Mecánica del automóvil
- Inversiones económicas
- Red de distribución de agua
- Venta de productos configurables
- Gestión de hospital (camas y servicios de guardias)
- Asesoría legal (laboral, fiscal, separación)

SE PIDE:

a) Para cada caso identificar cuál es el sistema de referencia y cuáles son sus características de acuerdo con la terminología utilizada en la descripción de sistemas.

b) Para cada caso identificar tipos de tareas que pueden llevarse a cabo.

3 Clasificación heurística

El método de clasificación heurística ha sido ampliamente utilizado en la construcción de sistemas inteligentes. Se trata de uno de los primeros métodos planteados como tal en el área de la ingeniería del conocimiento. Su definición se debe a Clancey [Clancey, 85] que, tras analizar diversos sistemas expertos existentes, mostró una estructura lógica común en ellos. Este método debe contemplarse como una familia de métodos, dado que admite diferentes estrategias de inferencia.

3.1 Descripción general

Para caracterizar el problema de clasificación de forma general se manejan los siguientes términos:

- *observaciones*: son valores observados o medidas sobre ciertas características del objeto a clasificar.

- *categorías*: son las diferentes categorías a las que un objeto puede pertenecer.

El problema de clasificación se define de la siguiente forma:

> **Definición 3.1:** Dado un conjunto de observaciones de un objeto, *clasificar* consiste en seleccionar en un conjunto prefijado de categorías la que mejor corresponde a dicho objeto.

En los problemas de clasificación el conjunto de categorías posibles se conoce a priori y es posible enumerarlo de forma explícita antes de tratar de resolver el problema. Un problema de tipo clasificativo será de *clasificación simple* si la solución es un único elemento del espacio de categorías. Cuando se admiten varias soluciones simultaneamente el problema será de *clasificación múltiple*.

Dentro del problema general de clasificación, además de problemas puramente clasificativos (por ejemplo, identificación de rocas, identificar un tipo de delito, etc.), se encuentran otros problemas habituales tales como:

- *Recomendar* una categoría de objeto en función de unos requisitos funcionales y preferencias (recomendar un tipo de deporte, recomendar un tipo de cámara fotográfica). Las observaciones en este caso corresponden a los deseos expresados sobre las funciones que debe cumplir el objeto a recomendar.

- *Evaluar* la idoneidad de un objeto para cumplir una determinada función prefijada (evaluar el cumplimiento de los requisitos de importación de un producto, evaluación de un candidato a un puesto de trabajo). En este caso, las categorías son las posibles decisiones acerca de la idoneidad.

- *Monitorizar* las desviaciones de un sistema dinámico (por ejemplo, vigilancia intensiva de un paciente, supervisión del comportamiento de una inversión diversificada). En este caso, las categorías solución son los diferentes tipos de desviaciones que se pueden detectar.

- *Diagnosticar* un sistema dinámico para encontrar las causas de unos síntomas dados (por ejemplo diagnosticar un paciente o el motor de un automóvil). En este caso las observaciones son los síntomas y las categorías son las posibles causas.

- *Predecir* el estado futuro de un sistema dinámico a partir de un estado actual y un conjunto de acciones. Las observaciones son el estado actual y las acciones. Las categorías son los diferentes estados futuros que puede presentar el sistema dinámico.

Las dificultades del problema de clasificación vienen dadas por los siguientes factores [Puppe, 93]:

- *Incertidumbre en el conocimiento de clasificación*. Para asociar las categorías a las observaciones se dispone de conocimiento sobre el problema que, bien por su complejidad o por desconocimiento, incluye simplificaciones y/o ausencia de criterios para tratar ciertas situaciones. Por ejemplo, en problemas como diagnóstico médico no se conocen siempre las razones últimas que relacionan síntomas con enfermedades aunque se manejan criterios en forma de cuadros clínicos con ciertos grados de seguridad parcial.

- *Observaciones poco fiables*. Las observaciones pueden presentar vaguedades o ser subjetivas (dolores en medicina, sensibilidad en la superficie, etc.). Además, las observaciones pueden presentar fallos, por ejemplo errores de medición de sensores.

- *Observaciones incompletas*. Por el coste de obtención de las observaciones (análisis clínicos, pruebas radiológicas, operaciones quirúrgicas, pruebas de medición de terrenos, etc.) se suele comenzar con un conjunto mínimo que se puede ir completando sucesivamente. Esto supone realizar el proceso de

clasificación llevando a cabo un diálogo con el usuario en donde el sistema solicita progresivamente información necesaria.

- *Manejo de pasos intermedios para abstracción*. No suele haber un camino directo de las observaciones a las categorías sino que se deben hacer pasos intermedios a lo largo de una cadena lógica en donde intervienen otro tipo de hechos a diferentes niveles de abstracción.

El método de *clasificación heurística* tiene dos características fundamentales:

1. El mecanismo de clasificación se basa en la presencia de relaciones heurísticas, es decir, relaciones establecidas mediante juicios subjetivos acumulados por la experiencia en la resolución de problemas (de ahí el nombre del método).

2. El proceso de clasificación maneja conceptos intermedios que corresponden a niveles de abstracción, lo cual reduce la complejidad del problema.

En el método de clasificación heurística, en vez de manejar relaciones directas entre observaciones y categorías, lo que significaría manejar normalmente un elevado número de opciones posibles, se maneja un nivel de abstracción que da lugar a dos espacios adicionales: un espacio de clases de observaciones y un espacio de clases de categorías. Así, las relaciones entre observaciones y categorías no se establecen de forma directa sino a través de dichos espacios intermedios, lo cual supone una simplificación importante dado que, normalmente, la dimensión de los espacios de clases de observaciones y clases de categorías es significativamente menor que la dimensión de los espacios iniciales.

Esto implica que en el proceso de clasificación deben contemplarse dos pasos de razonamiento adicionales: un paso de abstracción para ir de las observaciones iniciales a las clases de observaciones y otro paso de refino para pasar de las clases

de categorías a las categorías finales. Así, la figura 3.1 muestra un resumen de los conjuntos manejados por este método con los tres pasos de razonamiento que deben contemplarse: (1) *abstraer* para ir de observaciones a clases de observaciones, (2) *asociar*, para pasar de clases de observaciones a clases de categorías y (3) *refinar* para pasar de clases de categorías a categorías.

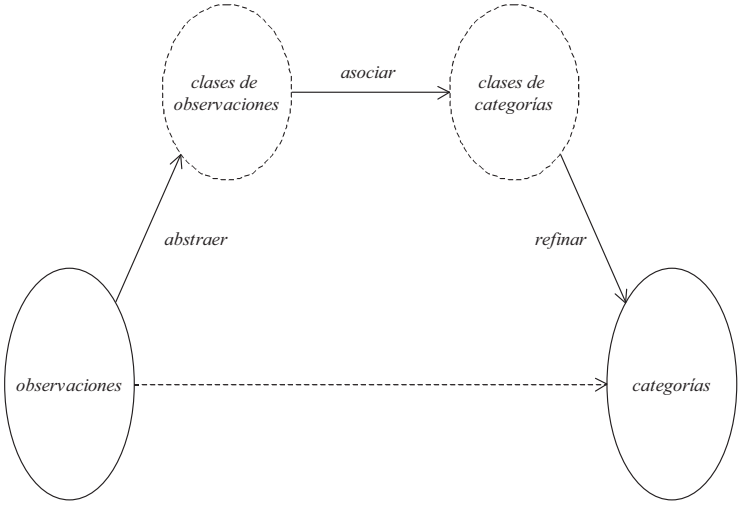


Figura 3.1: Pasos intermedios en el proceso de clasificación mediante el método de clasificación heurística.

3.2 Organización del conocimiento

De acuerdo con la descripción anterior, el conocimiento del método de clasificación se divide en tres bases de conocimiento, una para cada paso de razonamiento identificado: criterios de abstracción, relaciones heurísticas y jerarquía de categorías.

La base de conocimiento de *criterios de abstracción* incluye conocimiento que permite abstraer las observaciones recibidas como dato. Esta abstracción tiene dos fines principales: (1) reducir la dimensión del espacio considerado como información de partida y (2) pasar del lenguaje del usuario del sistema, que utiliza términos más coloquiales y de sentido común, al lenguaje del dominio profesional

en donde se resuelve el problema, que utiliza términos más técnicos y precisos. Los tipos de abstracciones que se pueden considerar son:

a) *Conocimiento de agregación cuantitativa*. Se trata de criterios expresados mediante formulación matemática que, a partir de los valores de variables primarias (por ejemplo, variables correspondientes a medidas de sensores), indican cómo calcular los valores de variables secundarias más representativas y cercanas a los procesos de decisión. Se trata de operaciones de naturaleza cuantitativa (estadística, aritmética, etc.). Ejemplos: (1) en tráfico, S = I/C (saturación S, intensidad I, capacidad C), (2) tendencia de los últimos valores medidos de un sensor de nivel en un embalse (3) agregaciones espaciales como la lluvia en una zona amplia a partir de la lluvia en diversos puntos, (4) agregaciones temporales, como la lluvia en la última hora a partir de medidas cada 15 minutos. Las representaciones que se pueden manejar en este caso son, por ejemplo, reglas o bien funciones numéricas para realizar la abstracción con un lenguaje declarativo que permita relacionar las diferentes variables. Aquí pueden incluirse también funciones más complejas para reconocimiento de señales que permitan una cierta forma de percepción de una realidad externa.

b) *Conocimiento para interpretación cualitativa*. Se utiliza para traducir una medida numérica a una etiqueta cualitativa con el fin de dar un cierto significado a dicha medida. Por ejemplo a partir de `temperatura(paciente) = 40` se determina `fiebre(paciente) = alta`, o bien, a partir de `recuento (leucocitos) = 2250` se determina `nivel(leucocitos) = bajo`. La representación en este caso puede ser basada en reglas y/o funciones de posibilidad de lógica difusa. Por ejemplo las reglas:

```
temperatura(paciente) > 39 → fiebre(paciente) = alta
temperatura(paciente) = [37, 39] → fiebre(paciente) = baja
temperatura(paciente) < 37 → fiebre(paciente) = nula
```

c) *Conocimiento de generalización*. Permite determinar la clase a la que pertenece una determinada observación. Por ejemplo, mediante dicho conocimiento es posible pasar de la observación correspondiente a la fecha del día de hoy `15-agosto-2004` a la estación del año correspondiente `estación = verano`. Al igual que los tipos de conocimiento anteriores, describe una agregación de casos que permite reducir el tamaño del espacio de datos (por ejemplo, reúne en un solo caso `estación = verano` todos los casos correspondientes a cada uno de los días de dicha estación). Para representar este conocimiento se puede utilizar una jerarquía con relaciones de tipo es-un o bien reglas tales como:

```
día >= 21 y mes = junio → estación = verano
mes = julio → estación = verano
mes = agosto → estación = verano
día < 21 y mes = septiembre → estación = verano
```

d) *Conocimiento sobre definiciones*. Indica la definición correspondiente a un determinado término. Es especialmente útil para traducir el lenguaje del usuario al lenguaje técnico del dominio en donde se resuelve el problema. Por ejemplo, en medicina se utilizan términos tales como `leucopenia` o `disnea` cuyo significado puede establecerse mediante reglas:

```
nivel(leucocitos) = bajo → síntoma(paciente) = leucopenia
sensación(respiración) = ahogo → síntoma(paciente) = disnea
```

La base de conocimiento de *relaciones heurísticas* incluye relaciones entre clases de observaciones y clases de categorías basada en juicios subjetivos acumulados por la experiencia en la resolución de problemas. Según Clancey, una relación heurística es incierta, basada en hipótesis de lo habitual y a veces sólo es una correlación pobremente entendida. Un heurístico es frecuentemente empírico, derivado de la experiencia en resolución de problemas. Las relaciones heurísticas reducen la búsqueda porque saltan relaciones intermedias.

Tipo/subtipo de conocimiento		**Explicación**	
Criterios de abstracción	Agregación cuantitativa	*Significado*	Permite componer valores de variables cuantitativas para obtener valores de variables cuantitativas más representativas
		Representación	Reglas, funciones (aritméticas, estadísticas, etc.)
		Ejemplos	`S= I/C, M = max (V1,V2,V3,V4)`
	Interpretación cualitativa	*Significado*	Relaciones entre intervalos numéricos y valores cualitativos
		Representación	Reglas, funciones de posibilidad (fuzzy)
		Ejemplos	`temperatura(paciente) > 39` → `fiebre(paciente) = alta`
	Generalización	*Significado*	Relaciones de generalización de tipo "es-un" entre hechos
		Representación	Reglas, atributos de conceptos, relaciones en marcos
		Ejemplos	`mes = agosto` → `estación = verano`
	Definiciones	*Significado*	Relaciones que expresan definiciones de hechos a partir de otros hechos
		Representación	Reglas
		Ejemplos	`nivel(leucocitos) = bajo` → `síntoma(paciente) = leucopenia`
Relaciones heurísticas		*Significado*	Relaciones entre clases de observaciones y clases de categorías, basada en juicios subjetivos derivados de la experiencia. Expresan condiciones que deben cumplirse sobre características que son parte de una categoría para que se pueda confirmar o rechazar.
		Representación	Reglas, marcos, medidas de incertidumbre
		Ejemplos	`tendencia (tipo-interés) = negativa` → `[0.8] recomendación(inversión) = bolsa`
Jerarquía de categorías		*Significado*	Estructura jerárquica del espacio de categorías en niveles de generalidad
		Representación	Reglas, atributos de conceptos, relaciones en marcos
		Ejemplos	`clase(deporte) = aire-libre,` `preferencia(individuo) = montaña` → `recomendación(deporte) = esquí`

Figura 3.2. Organización del conocimiento.

Por ejemplo, un determinado tipo de sonoridad del motor hace sospechar a un experimentado mecánico que el carburador está estropeado o un determinado conjunto de síntomas permite sospechar al médico que el paciente tiene una enfermedad. En el dominio de economía se puede manejar la siguiente relación heurística `tendencia (tipo-interés) = negativa → [0.8] recomendación (inversión) = bolsa` que se caracteriza porque:

- No presenta de forma explícita la cadena causal lógica entre antecedente y consecuente, dado que o bien no se conoce exactamente o es demasiado detallada para el nivel de representación que se requiere, lo que podría un proceso de búsqueda demasiado costoso. El ejemplo corresponde a la simplificación de un razonamiento causal detallado como el siguiente: bajan tipo de intereses → se reducen beneficios de inversiones en renta fija → búsqueda de inversiones alternativas → incremento de la demanda de acciones => tendencia ascendente de la bolsa → recomendable invertir en bolsa.

- Suele presentar medidas de confianza que pueden establecerse de forma subjetiva o se pueden calcular mediante tratamiento estadístico de casos. En el ejemplo este valor se muestra con la medida 0.8 como valor de certeza entre –1 y +1, siguiendo una forma de representación como la del método Mycin.

El conocimiento sobre relaciones heurísticas es en ocasiones incompleto debido a la falta de conocimiento en el dominio considerado. No obstante, permite llegar a un rendimiento similar al de las personas y su almacenamiento explícito posibilita la incorporación gradual de nuevo conocimiento conforme se va identificando a medida que se resuelven nuevos problemas. Por ello es muy importante que el sistema final proporcione explicaciones sobre qué conocimiento utiliza para alcanzar sus conclusiones con objeto de facilitar su puesta punto y calibración.

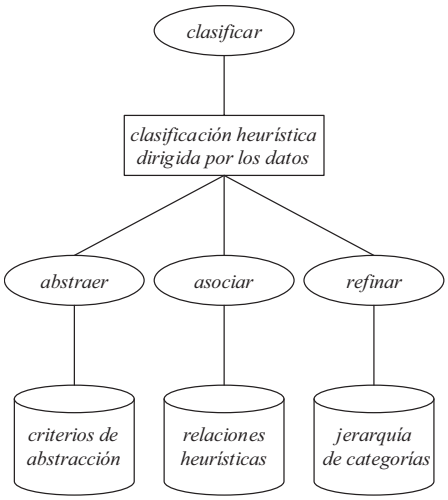


Figura 3.3: Estructura del método considerada en el algoritmo 1.

La representación utilizada habitualmente es mediante reglas y, opcionalmente, con medidas de incertidumbre. Las relaciones heurísticas se basan, en cierta forma, en un tipo de relación de naturaleza *es-parte-de* dado que para cada categoría indican condiciones que deben cumplirse sobre las características que la componen. Por ello, otra representación que puede utilizarse para este tipo de relaciones es con un conjunto de marcos en donde cada marco es una categoría y, para cada uno, se tienen *slots* que sobre observaciones y clases de observaciones que se utilizan mediante un procedimiento de equiparación para seleccionar uno o varios marcos como solución .

La base de conocimiento de *jerarquía de categorías* es una estructura jerárquica del espacio de categorías en niveles de generalidad (relaciones de tipo *es-un*) que permite detallar las clases de categorías hacia las categorías concretas. Una forma de representar este conocimiento es mediante reglas, en cuyos antecedentes se incluyen clases de categorías junto con observaciones adicionales y en los consecuentes los subtipos de categorías. Por ejemplo:

```
observaciones-1 → categoria-1
categoria-1, observaciones-2 → categoria-2
categoría-2, observaciones-3 → categoría-3
```

Como alternativa a esta representación, con el fin de separar mejor las relaciones heurísticas de la jerarquía, se puede manejar por un lado la jerarquía de soluciones como un grafo o reglas que simplemente conectan categorías con sub-categorías. Por ejemplo:

```
categoria-1 → categoría-2
categoria-1 → categoria-3
```

Y, por otro lado, reunido en el conocimiento de relaciones heurísticas, se indican las relaciones (en forma de reglas por ejemplo) que indican únicamente las condiciones que permiten confirmar las categorías, asumiendo que ya han sido confirmadas las categorías de nivel superior. Por ejemplo:

```
observaciones-1 → categoria-1
observaciones-2 → categoria-2
observaciones-3 → categoria-3
```

3.3 Inferencia

Para el método de clasificación heurística se pueden considerar varias opciones posibles de estrategias de inferencia: (1) dirigida por los datos, que va de las observaciones a las categorías, (2) dirigida por los objetivos, que va de las categorías a las observaciones, (3) clasificación jerárquica, que también va de las categorías a las observaciones pero analizando las categorías en un proceso de

búsqueda jerárquica en vez de lineal y (4) clasificación jerárquica dirigida por los datos, que supone una variante del caso anterior iniciando la búsqueda desde nodos intermedios. Los siguientes apartados describen cada una de estas opciones.

Rol dinámico	Significado	Representación	Ejemplo
observaciones	datos observados utilizados como información de entrada	conjunto de ternas concepto-atributo-valor	{temperatura(paciente)=40, edad(paciente)=40, sexo(paciente)=varón}
clases-observaciones	abstracciones de observaciones	conjunto de ternas concepto-atributo-valor	{fiebre(paciente)=alta, nivel-edad(paciente)=adulto}
clases-categorías	abstracciones de categorías	conjunto de ternas concepto-atributo-valor	{tipo(organismo)=gram-positivo, localización(organismo)=pulmón}
categorías	soluciones del problema de clasificación	conjunto de ternas concepto-atributo-valor	{nombre(organismo)=E-coli}

Figura 3.4: Roles dinámicos que intervienen en el proceso de inferencia del algoritmo 1.

```
INFERENCIA abstraer
       DATOS:                 observaciones
       BASES DE CONOCIMIENTO: criterios-de-abstracción
       RESULTADOS:            clases-observaciones

INFERENCIA asociar
       DATOS:                 observaciones, clases-observaciones
       BASES DE CONOCIMIENTO: relaciones-heurísticas
       RESULTADOS:            clases-categorías

INFERENCIA refinar
       DATOS:                 observaciones, clases-observaciones, clases-categorías
       BASES DE CONOCIMIENTO: jerarquía-de-categorías
       RESULTADOS:            categorías
```

Figura 3.5: Pasos de inferencia considerados en el algoritmo 1.

3.3.1 Algoritmo 1: Clasificación heurística dirigida por los datos

El algoritmo con estrategia dirigida por los datos corresponde a un procedimiento relativamente sencillo que asume que se dispone a priori (antes de comenzar la resolución del problema) de todos los datos sobre observaciones. Esta característica hace que su utilidad sea reducida dado que hay un gran número de problemas de clasificación en donde la información sobre observaciones se suministra gradualmente según se resuelve el problema lo que permite centrarse sólo en los datos que son relevantes en cada caso. El método se presenta aquí fundamentalmente con el fin de mostrar con un caso sencillo la notación utilizada en la descripción de los métodos, además de servir como base para una mejor comprensión de los métodos que se presentan después. El método maneja tres pasos de inferencia: abstraer, asociar y refinar. El paso de inferencia *abstraer* toma como entrada los datos recibidos como observaciones y, utilizando la base de conocimiento de criterios de abstracción, genera clases de observaciones. Típicamente, se puede implementar mediante un encadenamiento hacia delante en una base de reglas o en una red de cálculos prefijados.

```
METODO clasificación-heurística-dirigida-por-los-datos
  DATOS: observaciones
  RESULTADOS: categorías

ALGORITMO
1. abstraer(observaciones -> clases-observaciones)
2. asociar(observaciones, clases-observaciones -> clases-categorías)
3. refinar(observaciones, clases-observaciones, clases-categorías -> categorías)
```

Figura 3.6: Algoritmo 1 para el método de clasificación heurística.

El paso de inferencia *asociar* recibe como entrada hechos referidos a observaciones y clases de observaciones, y genera clases de categorías, utilizando la base de conocimiento las relaciones de heurísticas. Este paso de inferencia podría

programarse por ejemplo mediante inferencia en reglas hacia delante o mediante un procedimiento de equiparación en marcos.

Finalmente, el paso de inferencia *refinar* permite pasar de clases de categorías, junto con información de observaciones (directa o de clases), a categorías específicas. Para implementar este paso de inferencia también se puede realizar un encadenamiento hacia delante en reglas.

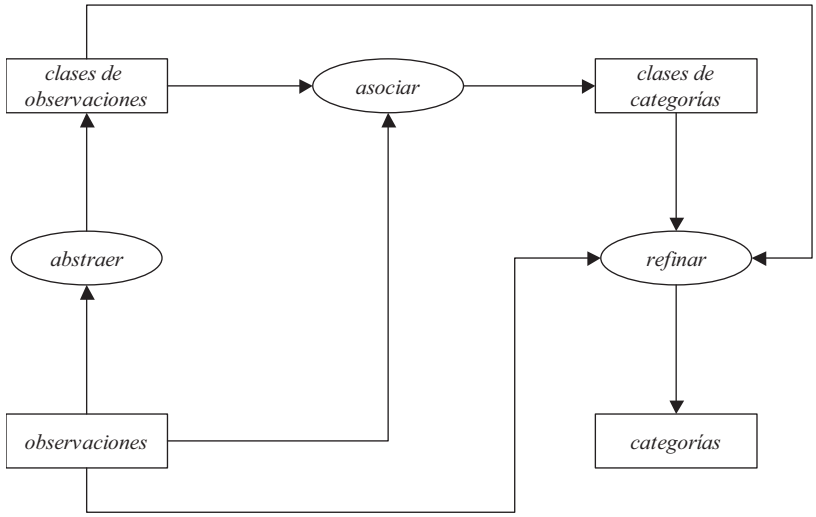


Figura 3.7: Estructura de inferencia para el algoritmo 1.

La limitación principal de este método, como ya se ha indicado, es que necesita todos los datos del problema antes de comenzar el razonamiento. Esto hace que no sea aplicable en situaciones en donde no se tienen todos los datos a priori debido a que son muchos y/o difíciles de conocer porque su obtención supone un determinado coste (análisis médicos, mediciones de terreno, etc.).

3.3.2 Algoritmo 2: Clasificación heurística dirigida por los objetivos

Este algoritmo corresponde a una versión similar al anterior pero con un mecanismo de control inverso, es decir, los pasos de razonamiento se realizan hacia atrás. Es adecuado cuando se manejan un número no muy elevado de objetivos y los datos se van obteniendo de forma progresiva (por ejemplo, mediante consulta al usuario). Básicamente esta versión consiste en generar inicialmente todas las hipótesis posibles de categorías y a continuación verificar una a una hacia atrás, preguntando la información que en cada momento sea necesario conocer para poder mantener o descartar cada hipótesis. Se manejan tres pasos de inferencia, que se corresponden en cierta forma con la versión hacia atrás de los pasos de inferencia presentados en la versión previa: generar-todas, probar y adquirir. El paso de inferencia *generar-todas* tiene como objetivo generar todas las hipótesis de categorías. Para ello, utiliza la base de conocimiento de jerarquía de categorías de donde extrae todas las categorías posibles (no necesita datos de entrada).

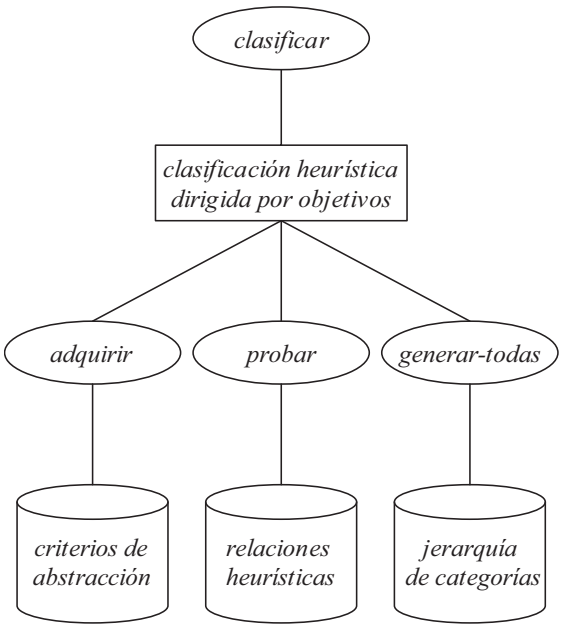


Figura 3.8: Estructura del método considerada en el algoritmo 2.

Rol dinámico	Significado	Representación	Ejemplo
observaciones	datos observados utilizados como información de entrada del problema de clasificación	conjunto de ternas concepto-atributo-valor	{temperatura(paciente)=40, sexo=varón(paciente), fiebre(paciente)=alta, nivel-edad(paciente)=adulto}
observaciones-requeridas	observaciones (o clases de observaciones) que deben ser consultadas al usuario	conjunto de pares concepto-atributo	{fiebre(paciente)}
conjunto-hipótesis	conjunto de hipótesis de categorías	conjunto de ternas concepto-atributo-valor	{nombre(organismo)=E-coli, nombre(organismo)=Pseudomona}
prueba	resultado de la prueba de la hipótesis	valor de {VERDADERO, FALSO, DESCONOCIDO}	VERDADERO
hipótesis	una hipótesis de categoría	terna concepto-atributo-valor	nombre(organismo)=E-coli
categorías	soluciones del problema de clasificación	conjunto de ternas concepto-atributo-valor	{nombre(organismo)=E-coli}

Figura 3.9: Roles dinámicos que intervienen en el proceso de inferencia del algoritmo 1.

El paso de inferencia *probar* recibe como entrada una hipótesis de categoría que se desea verificar junto con las observaciones que se tienen hasta ese momento. Como resultado, se contesta si es posible confirmar o descartar la hipótesis a partir de las observaciones utilizando el conocimiento de relaciones heurísticas. Esta información de salida se indica en el rol dinámico *prueba* que tomará valor VERDADERO en caso de que se confirme la hipótesis o FALSO en caso de que se rechace. Un hecho puede rechazarse en una base de reglas, por ejemplo, si se tienen reglas que niegan la conclusión. Se contempla además el valor DESCONOCIDO para cuando no es posible pronunciarse a favor o en contra. En ese caso, además, se pueden indicar en *observaciones-requeridas* los atributos cuyo valor debería conocerse para poder continuar con el proceso demostrativo que permita verificar o rechazar la hipótesis.

```
INFERENCIA generar-todas
        DATOS:                  -
        BASES DE CONOCIMIENTO: jerarquía-de-categorías
        RESULTADOS:             conjunto-hipótesis

INFERENCIA probar
        DATOS:                  hipótesis, observaciones
        BASES DE CONOCIMIENTO: relaciones-heurísticas
        RESULTADOS:             prueba, observaciones-requeridas

INFERENCIA adquirir
        DATOS:                  observaciones-requeridas, observaciones
        BASES DE CONOCIMIENTO: criterios-de-abstracción
        RESULTADOS:             observaciones
```

Figura 3.10: Pasos de inferencia considerados en el algoritmo 2.

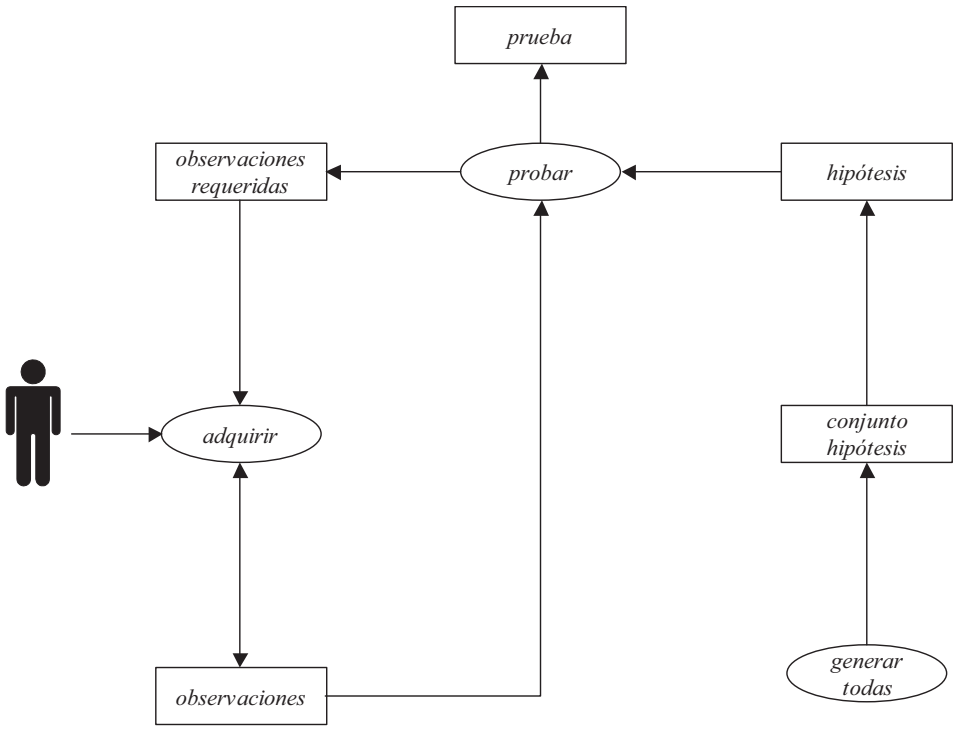


Figura 3.11: Estructura de inferencia del algoritmo 2.

Por ejemplo, supóngase que la hipótesis que se desea probar es H=a y se tienen las siguientes reglas:

A=a y B=b → E=a

E=a → H=a

C=a y D=a → H=a

A=b → NO(H=a)

C=b → NO(H=a)

Si el conjunto de observaciones es {A=a, B=b} la respuesta es VERDADERO. Si el conjunto es {A=b} la respuesta es FALSO. Si se tiene {A=a, B=a} entonces la respuesta es DESCONOCIDO y será necesario conocer el valor de C que se incluye en el conjunto de observaciones requeridas.

```
METODO clasificacion-heurística-dirigida-por-los-objetivos
  DATOS: observaciones
  RESULTADOS: categorías

ALGORITMO
1. generar-todas( -> conjunto-hipótesis)
2. FOR-EACH hipótesis IN conjunto-hipótesis DO
3.   REPEAT
4.      probar(hipótesis, observaciones -> prueba, observaciones-requeridas)
5.      IF prueba = VERDADERO
6.      THEN categorías := categorías U {hipótesis}
7.      ELSE
8.        IF NOT(observaciones-requeridas = ϕ)
9.       THEN adquirir(observaciones-requeridas, observaciones ->  observaciones)
10.   UNTIL observaciones-requeridas = ϕ
```

Figura 3.12: Algoritmo 2 para el método de clasificación heurística.

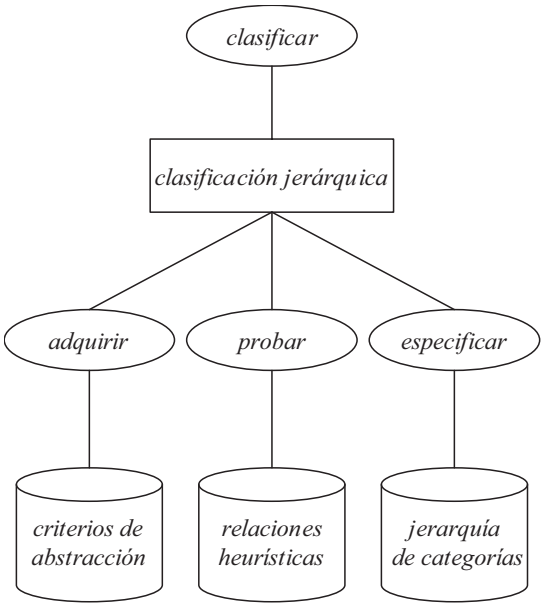


Figura 3.13: Estructura del método considerada en el algoritmo 3.

El paso de inferencia *adquirir* toma como entrada las observaciones-requeridas, que incluye los atributos cuyo valor debería conocerse para poder verificar o descartar la hipótesis. Utiliza el conocimiento de abstracción para determinar las preguntas que deben hacerse al usuario. Esto permite separar el lenguaje en el que se le interroga al usuario, que será en general menos especializado y estará basado en observaciones objetivas, del lenguaje interno del sistema que utiliza términos más técnicos con mayor nivel de abstracción. El contenido del conjunto de observaciones aumenta con las observaciones resultantes debidas a las respuestas del usuario, además de las deducidas mediante los criterios de abstracción.

Por ejemplo como entrada a este paso de inferencia se puede tener `observaciones-requeridas = {ictericia(paciente)}` que indica que debería conocerse el atributo ictericia del paciente para poder continuar con el proceso de verificación de la hipótesis en curso. El paso de inferencia analiza del conocimiento de abstracción que permite deducir el valor de dicho atributo. Por ejemplo, si se manejasen reglas, se puede tener una regla que dice `nivel (bilirrubina) > 10` → `ictericia`

`(paciente) = sí`. Entonces el sistema pregunta al usuario el valor de bilirrubina (medida que se encuentra un análisis de sangre). Si el usuario contesta por ejemplo `nivel(bilirrubina) = 11.7` entonces se deduce además `ictericia(paciente) = sí`, pasando a formar parte ambos hechos del conjunto de observaciones.

Si el usuario hubiera respondido que no conoce el valor del atributo, entonces se intenta deducir el valor de ictericia con otras reglas. Si finalmente no es posible deducir dicho valor, en el conjunto de observaciones se almacena `nivel(bilirrubina) = desconocido`, `ictericia (paciente) = desconocido`. Al incluir esta anotación en el conjunto de observaciones se evita que el paso de inferencia *probar* incluya de nuevo en el conjunto de observaciones requeridas el atributo correspondiente a ictericia.

```
INFERENCIA especificar
       DATOS:                 hipótesis
       BASES DE CONOCIMIENTO: jerarquía-de-categorías
       RESULTADOS:            conjunto-hipótesis

INFERENCIA probar
       DATOS:                 hipótesis, observaciones
       BASES DE CONOCIMIENTO: relaciones-heurísticas
       RESULTADOS:            prueba, observaciones-requeridas

INFERENCIA adquirir
       DATOS:                 observaciones-requeridas, observaciones
       BASES DE CONOCIMIENTO: criterios-de-abstracción
       RESULTADOS:            observaciones
```

Figura 3.14: Pasos de inferencia considerados en el algoritmo 3.

En esta versión del algoritmo se asume por simplicidad que *observaciones-requeridas* tiene únicamente el primer atributo que es necesario conocer. En versiones más elaboradas se podrían incluir expresiones lógicas, por ejemplo una conjunción de condiciones sobre valores de atributos. El proceso de *adquirir* en ese

caso debería ser capaz de tratar expresiones lógicas de forma que, si es una conjunción, intentaría determinar el valor del primer atributo y, si el valor satisface la condición, se continuaría con el siguiente atributo pero si no satisface la condición el proceso se detiene.

3.3.3 Algoritmo 3: Clasificación jerárquica

Cuando hay un número elevado de categorías no es eficiente hacer un recorrido completo de todas ellas para verificar si se corresponden con las observaciones, tal como se hace en el algoritmo anterior. Por ello se plantea la siguiente mejora denominada *clasificación jerárquica* (o también *establecer y refinar*) que recorre las categorías descendiendo en un árbol de hipótesis organizadas en niveles de generalidad (con relaciones de tipo *es-un*). La búsqueda comienza desde la hipótesis raíz y se va profundizando en el árbol a medida que se verifica cada una, de forma que no se comprueban las categorías descendientes de las clases descartadas. En este proceso se manejan los mismos pasos de inferencia que en el algoritmo anterior, excepto el paso de inferencia que se encarga de generar las hipótesis que, en vez de generar el conjunto completo de hipótesis de una sola vez, las va generando de forma progresiva haciendo un recorrido hacia los nodos hoja en la jerarquía de categorías.

Así, el paso de inferencia *especificar* recibe como entrada una hipótesis de categoría y genera el conjunto de subhipótesis que son descendientes inmediatas en la jerarquía de categorías. Esto puede implementarse de forma sencilla haciendo uso de un grafo en forma de árbol y un proceso que devuelve los nodos descendientes de uno dado. Los otros dos pasos coinciden con los pasos descritos en el algoritmo del apartado anterior. Los roles dinámicos son también los mismos que se manejan en la versión anterior.

```
METODO clasificación-jerárquica
  DATOS: observaciones
  RESULTADOS: categorías

ALGORITMO
1. hipótesis := nodo raíz de la jerarquía de categorías
2. categorías := ϕ
3. establecer(hipótesis, categorías, observaciones -> categorías, observaciones)

PROCEDIMIENTO establecer
  DATOS: hipótesis, categorías, observaciones
  RESULTADOS: categorías, observaciones
1. REPEAT
2.   probar(hipótesis, observaciones -> prueba, observaciones-requeridas)
3.   IF prueba = VERDADERO
4.   THEN
5.     refinar(hipótesis, categorías, observaciones -> categorías, observaciones)
4.   ELSE
5.     IF NOT(observaciones-requeridas = ϕ)
6.     THEN adquirir(observaciones-requeridas, observaciones -> observaciones)
7. UNTIL observaciones-requeridas = ϕ

PROCEDIMIENTO refinar
  DATOS: hipótesis, categorías, observaciones
  RESULTADOS: categorías, observaciones
1. especificar(hipótesis -> conjunto-hipótesis)
2. IF conjunto-hipótesis = ϕ
3. THEN categorías := categorías U {hipótesis}
4. ELSE FOR-EACH hipótesis IN conjunto-hipótesis DO
5.   establecer (hipótesis, categorías, observaciones -> categorías, observaciones)
```

Figura 3.15: Algoritmo 3 (clasificación jerárquica) para clasificación heurística.

La figura 3.15 muestra el algoritmo considerado, formado por un cuerpo principal y dos procedimientos que se invocan el uno al otro, realizando el descenso en la jerarquía mediante un proceso recursivo. El primero de los procedimientos tiene como fin confirmar una hipótesis (establecer) haciendo uso de los pasos de inferencia probar y adquirir. El segundo es el encargado de obtener subhipótesis (refinar) y continuar el proceso. Se manejan dos conjuntos denominados observaciones y categorías cuyo contenido puede ir creciendo progresivamente a lo

largo del proceso y contienen respectivamente (1) las observaciones dadas por el usuario y deducidas y (2) las categorías solución que se van encontrando.

En esta versión de algoritmo de clasificación jerárquica se profundiza en los nodos de la jerarquía de soluciones únicamente cuando se confirma una hipótesis, es decir, cuando la respuesta del paso de inferencia es VERDADERO. Este enfoque está asumiendo implícitamente la simplificación de *hipótesis de mundo cerrado*, es decir, cuando no es posible deducir un hecho se asume que es falso. En dominios donde no se desee realizar dicha simplificación puede fácilmente modificarse el algoritmo de forma que se contemple también el caso DESCONOCIDO profundizando por dichos nodos, por ejemplo, después de haber realizado la búsqueda por los nodos que resultaron confirmados.

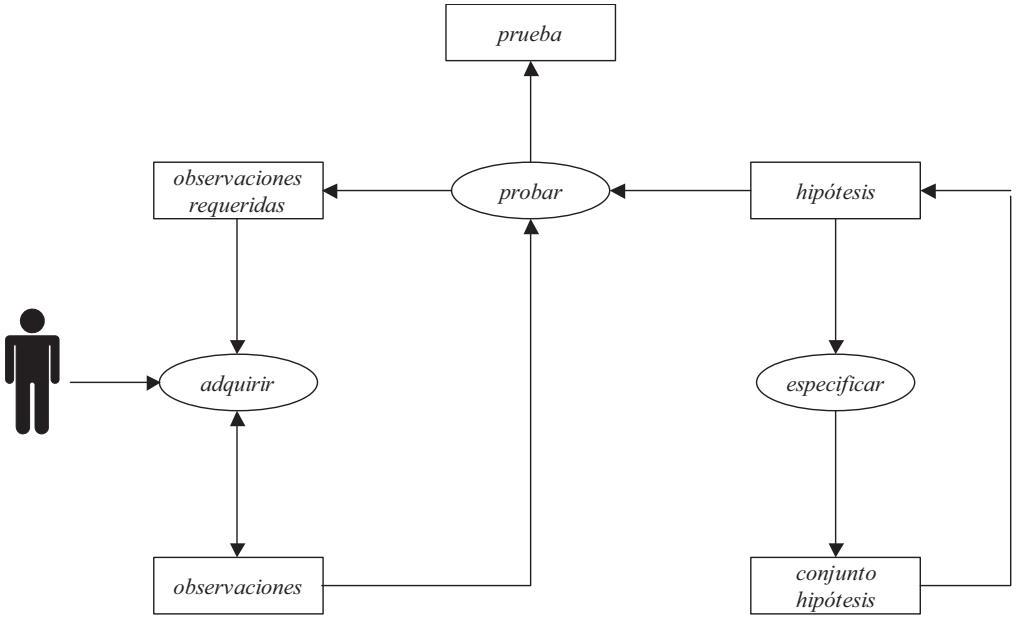


Figura 3.16: Estructura de inferencia del algoritmo 3.

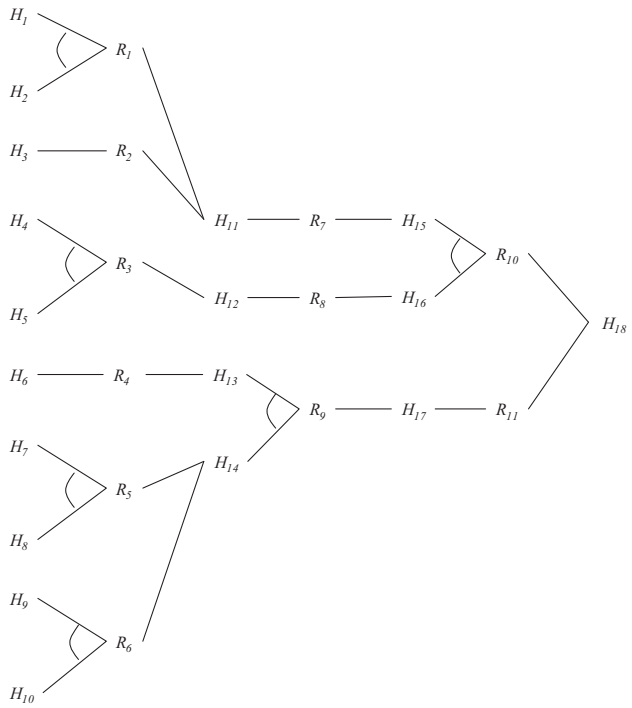


Figura 3.17: Gráfico utilizado para ejemplo del paso de inferencia para sugerir hipótesis

3.3.4. Algoritmo 4: Clasificación jerárquica dirigida por los datos

El algoritmo anterior tiene como limitación que realiza una navegación sistemática por la jerarquía partiendo de la hipótesis raíz. El comienzo obligado por la raíz de la jerarquía puede no ser realista en ciertas ocasiones cuando los datos iniciales permiten plantear hipótesis correspondientes a nodos intermedios. Para ello se plantea aquí una variante que incluye la posibilidad de aprovechar las observaciones iniciales que permitan seleccionar unas hipótesis intermedias desde donde comienza la búsqueda hacia los nodos inferiores. La forma de seleccionar dichas hipótesis intermedias es mediante un encadenamiento eficiente desde las observaciones hasta las categorías reutilizando el conocimiento de abstracción y de relaciones heurísticas en la dirección inversa al que se realiza en el método general de clasificación jerárquica. Dicho encadenamiento se realiza de forma parcial hacia las

categorías, teniendo en cuenta que se plantean hipótesis de categorías que *generan* las observaciones. En este contexto, el término *generar* se define de la siguiente forma:

> **Definición 3.2:** Una observación *genera* una hipótesis de categoría si dicha observación pertenece al conjunto total de premisas a partir de las cuales es posible deducir la categoría.

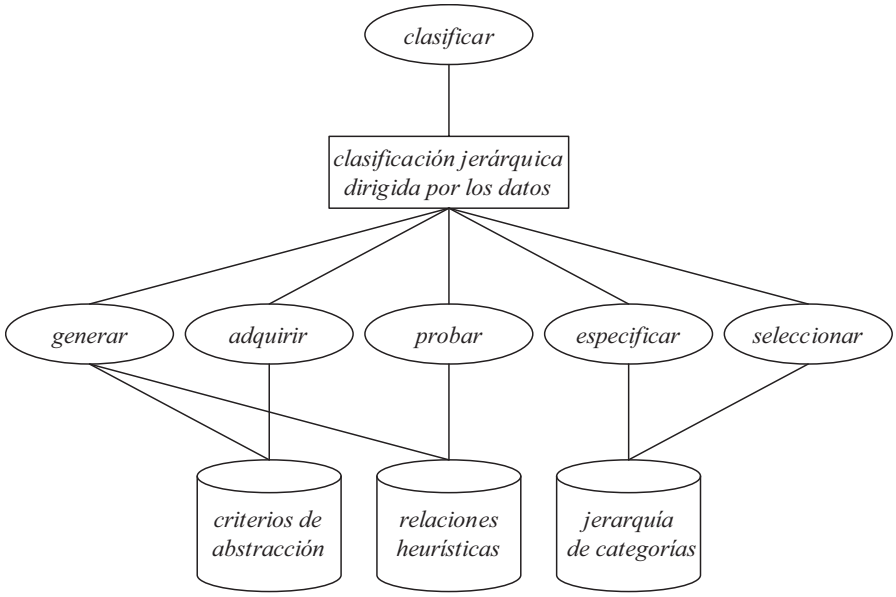


Figura 3.18: Estructura del método considerada en el algoritmo 4.

Esta idea se ilustra en el ejemplo mostrado en la figura 3.17. La figura muestra un árbol deductivo que relaciona una serie de hechos {Hi} con un conjunto de reglas {Rj}. Los hechos de más a la izquierda corresponden a las observaciones y el de más a la derecha es una categoría. Las reglas corresponden al conocimiento de relaciones de abstracción y heurísticas, por ejemplo los criterios de abstracción están en las reglas {R1, ..., R7} y las relaciones heurísticas en las reglas {R8, ...,

R11}. El conjunto total de premisas que permiten deducir la categoría son las hojas del árbol {H1, ..., H10}. En este caso, basta con que se observe alguna de las hojas de dicho árbol, por ejemplo H5, para que se plantee como posible la hipótesis H18, es decir, se genere H18. En un proceso posterior se determinará si realmente es posible deducir H18 únicamente con H5 o, si por el contrario, es necesario disponer de información sobre otros hechos. Este proceso se puede realizar de forma muy eficiente si se realiza un tratamiento previo de las reglas, antes de realizar las sucesivas consultas, construyendo una estructura de datos adicional que asocia a cada categoría las observaciones que la sugieren.

```
INFERENCIA generar
      DATOS:                 observaciones
      BASES DE CONOCIMIENTO: criterios-de-abstracción, relaciones-heurísticas
      RESULTADOS:            candidatas

INFERENCIA seleccionar
      DATOS:                 candidatas, revisadas
      BASES DE CONOCIMIENTO: jerarquía-de-categorías
      RESULTADOS:            hipótesis

INFERENCIA especificar
      DATOS:                 hipótesis
      BASES DE CONOCIMIENTO: jerarquía-de-categorías
      RESULTADOS:            candidatas

INFERENCIA probar
      DATOS:                 hipótesis, observaciones
      BASES DE CONOCIMIENTO: relaciones-de-asociación
      RESULTADOS:            prueba,  observaciones-requeridas

INFERENCIA adquirir
      DATOS:                 observaciones-requeridas, observaciones
      BASES DE CONOCIMIENTO: relaciones-de-abstracción
      RESULTADOS:            observaciones
```

Figura 3.19: Pasos de inferencia considerados en el algoritmo 4.

En esta versión se manejan 5 pasos de inferencia. Se mantienen los mismos tres pasos que se manejan en el algoritmo anterior (especificar, probar y adquirir) y se

incluyen dos más: generar y seleccionar. El paso de *generar* tiene como fin generar todas las hipótesis a partir de un conjunto de observaciones según el proceso descrito anteriormente. El paso de *seleccionar* se utiliza para elegir una hipótesis entre las candidatas. Una forma de hacer este proceso es elegir primero la hipótesis de nivel más específico en la jerarquía, para lo que se utiliza el conocimiento de jerarquías de categorías. Por conveniencia sobre la forma de escribir el algoritmo, este paso de inferencia recibe como entrada adicional las hipótesis ya revisadas que no se tendrán en cuenta en el proceso de selección. Por ejemplo, considérese el ejemplo de la figura 3.20. En este ejemplo el resultado de *generar* daría lugar a `candidatas = {H1, H2, H3}`. El proceso de seleccionar elige primero para realizar la búsqueda a H2 anotando dicha hipótesis en `revisadas = {H2}`. Después, si la búsqueda por H2 no tiene éxito y se mantienen las mismas hipótesis se elegiría H3.

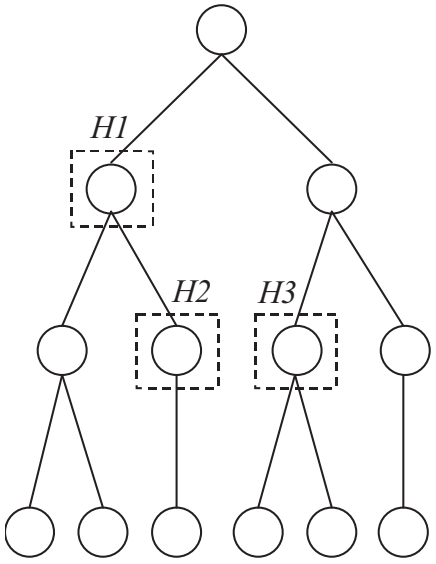


Figura 3.20: Ejemplo de selección de hipótesis.

Rol dinámico	Significado	Representación	Ejemplo
observaciones	datos observados utilizados como información de entrada del problema de clasificación	conjunto de ternas concepto-atributo-valor	{temperatura(paciente)=40, sexo(paciente)=varón, fiebre(paciente)=alta, nivel-edad(paciente)=adulto}
observaciones-requeridas	observaciones (o clases de observaciones) que deben ser consultadas al usuario	conjunto de pares concepto-atributo	{fiebre(paciente)}
hipótesis	una hipótesis de categoría	terna concepto-atributo-valor	nombre(organismo)=E-coli
candidatas	conjunto de hipótesis de categoría candidatas	conjunto de ternas concepto-atributo-valor	{nombre(organismo)=E-coli, nombre(organismo) =pseudomona}
revisadas	conjunto de hipótesis que han sido ya estudiadas y que no deben considerarse candidatas	conjunto de ternas concepto-atributo-valor	{nombre(organismo)=E-coli, nombre(organismo)= pseudomona}
prueba	resultado de la prueba de la hipótesis	valor de {VERDADERO, FALSO, DESCONOCIDO}	VERDADERO
éxito	indica si ha tenido éxito la búsqueda de la solución	valor de {VERDADERO, FALSO}	VERDADERO
categoría	solución del problema de clasificación	terna concepto-atributo-valor	nombre(organismo)=E-coli

Figura 3.21: Roles dinámicos que intervienen en el proceso de inferencia.

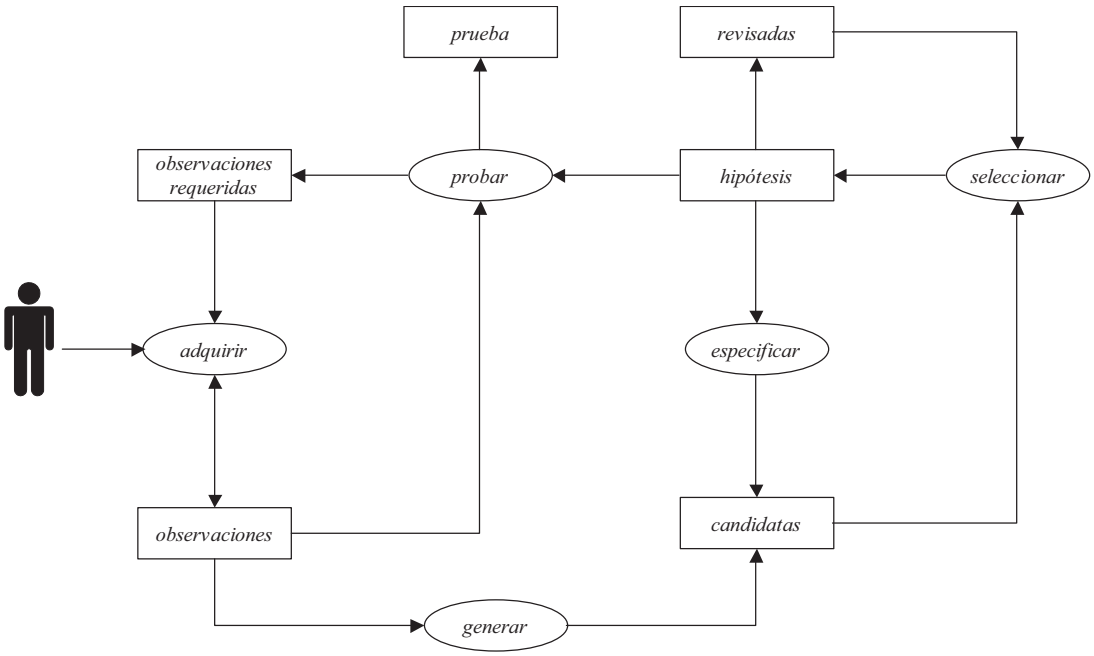


Figura 3.22: Estructura de inferencia del ejemplo de algoritmo 4.

```
METODO clasificacion-jerárquica-dirigida-por-datos
DATOS: observaciones
RESULTADOS: categoría, éxito

ALGORITMO
1.   revisadas := ϕ
2.   generar(observaciones -> candidatas)
3.   seleccionar(candidatas, revisadas -> hipótesis)
4.   establecer(hipótesis, observaciones, revisadas ->
5.              categoría, observaciones, revisadas, éxito)
```

Figura 3.23: Cuerpo principal del algoritmo 4.

```
PROCEDIMIENTO establecer
  DATOS: hipótesis, observaciones, revisadas
  RESULTADOS: categoría, observaciones, revisadas, éxito
1.   revisadas := revisadas U {hipótesis}
2.   REPEAT
3.     probar(hipótesis, observaciones ->  prueba, observaciones-requeridas)
4.     IF prueba = VERDADERO
5.     THEN
6.       refinar(hipótesis, observaciones, revisadas ->
7.               categoría, observaciones, revisadas, éxito)
8.     ELSE
9.       IF NOT(observaciones-requeridas = ϕ)
10.      THEN
11.        adquirir(observaciones-requeridas, observaciones -> observaciones)
12.  UNTIL observaciones-requeridas = ϕ
13.  IF (prueba = FALSO) OR (prueba = DESCONOCIDO) THEN éxito = FALSO

PROCEDIMIENTO refinar
  DATOS: hipótesis, observaciones, revisadas
  RESULTADOS: categoría, observaciones, revisadas, éxito
1.   especificar(hipótesis -> candidatas)
2.   IF candidatas = ϕ
3.   THEN
4.     categoría := hipótesis
5.     éxito := VERDADERO
6.   ELSE
7.     generar(observaciones -> H)
8.     candidatas := candidatas U H
9.     REPEAT
10.      seleccionar(candidatas, revisadas -> hipótesis)
11.      establecer(hipótesis, observaciones, revisadas ->
12.                 categoría, observaciones, revisadas, éxito)
13.    UNTIL (éxito = VERDADERO) OR (candidatas SUBSET-OF revisadas)
```

Figura 3.24: Procedimientos del algoritmo 4.

Las figuras 3.23 y 3.24 muestran en pseudo-código el ejemplo de algoritmo 4. El algoritmo es similar a la versión de clasificación jerárquica general con dos procedimientos que operan de forma recursiva, incluyendo además las llamadas a los pasos de inferencia adicionales que se plantean en esta versión.

Aspectos de búsqueda y eficiencia

En resumen, los algoritmos presentados suponen una evolución de la versión más sencilla del algoritmo 1 hasta las más compleja y realista del algoritmo 4 que tiene un gran potencial de aplicabilidad en problemas de clasificación de dimensión real. No obstante, estos métodos pueden ser adaptados y extendidos en cada realización particular, por ejemplo, para considerar el coste de adquisición de observaciones (por ej., el coste de análisis y pruebas diagnósticas a un paciente). De forma resumida las opciones de búsqueda que se pueden contemplar son:

a) Recorrido de hipótesis de una en una. Esto puede realizarse (1) con simplificación de mundo cerrado en donde sólo se consideran las hipótesis que se prueban con resultado VERDADERO (versión seguida en el algoritmo del presente texto) o (2) sin esta simplificación por lo que se consideran también las que se prueban con resultado DESCONOCIDO aunque su análisis se efectúa cuando no hay hipótesis confirmadas.

b) Recorrido de hipótesis de una en una asumiendo que las hipótesis de un mismo nivel son excluyentes. En ese caso, cuando se confirma una, se pueden descartar el resto del mismo nivel. Como alternativa a este enfoque se puede contemplar exclusión parcial, es decir, cuando se confirma una hipótesis se descartan algunas (no todas las restantes) de su mismo nivel. La forma en que unas hipótesis excluyen otras se indica explícitamente en una base de conocimiento adicional.

c) Tratamiento conjunto de hipótesis del mismo nivel estudiando cuál es la observación que mejor discrimina entre todas con menor coste de adquisición. Para ello, se debe incluir una base de conocimiento adicional con el coste de las observaciones.

El problema de clasificación múltiple de forma general es intratable (NP-hard) tal como fue demostrado para el caso de razonamiento abductivo por Tom Bylander del grupo de Chandrasekaran [Bylander et al., 91] debido a la necesidad de generación de combinaciones de hipótesis de categoría que deben estudiarse conjuntamente para determinar la mejor teniendo en cuenta, además, los efectos de cancelación (el número de combinaciones crece de forma exponencial con el número de hipótesis existentes). Por ejemplo, en medicina una enfermedad puede explicar un exceso de pH y otra enfermedad un decremento de pH, pero si se dan simultaneamente pueden dar lugar a un pH normal lo cual cancela la presencia de observaciones. Una forma de enfrentarse a esta dificultad puede ser redefiniendo el problema de clasificación general. Por ejemplo, en vez de manejar la estrategia de maximizar plausibilidad (es decir, dados unos datos a ser explicados, se busca el conjunto de hipótesis más plausible, coherente, mínimo que explica *todos* los datos) que es computacionalmente intratable de forma general, se maneja una estrategia de maximizar el cubrimiento explicativo, que sí es computacionalmente tratable (dados los datos a ser explicados, se busca un conjunto de hipótesis coherente, mínimo, verosímil, que explica *la mayor parte* de los datos) [Josephson et al., 96].

3.4 Ejemplos

Se presentan a continuación varios ejemplos ilustrativos de clasificación heurística. El primero corresponde al caso del sistema Mycin y tiene como objetivo mostrar con un ejemplo real los diversos tipos de conocimiento que maneja el método. El segundo ejemplo se incluye para ilustrar el proceso de razonamiento del método (algoritmo 3) para un caso simplificado inspirado en el sistema MDX. Finalmente se muestra un ejemplo que analiza la relación de este método con los árboles de decisión. Otro ejemplo de clasificación heurística no presentado en este capítulo es la herramienta MORE, una herramienta de adquisición del conocimiento para construir sistemas con el método de clasificación heurística, que fue aplicada al dominio de vertido de fluidos contaminantes [Kahn, 88].

3.4.1. Ejemplo 1: Identificación de gérmenes bacterianos

Se describe aquí un ejemplo de clasificación en el campo de gérmenes bacterianos correspondiente al dominio en donde operaba el sistema Mycin. Aunque originalmente dicho sistema no se construyó teniendo en cuenta el método de clasificación heurística, por su forma se puede considerar encuadrado en dicho método y, de hecho, fue uno de los sistemas de referencia que utilizó Clancey para identificar la estructura de inferencia de clasificación heurística. Este sistema fue desarrollado dentro de un proyecto de investigación iniciado en 1972 en la Universidad de Stanford por Bruce G. Buchanan y Edward H. Shortliffe [Buchanan, Shortliffe, 84] y se considera uno de los pioneros de la arquitectura basada en el conocimiento. El nombre Mycin fue propuesto por Staton Axline, uno de los expertos en medicina que trabajaron en el sistema. El nombre se deriva del sufijo común que presentan los nombres de algunos de antibióticos por ejemplo *Erythromycin* o *Clindamycin* (Eritromicina y Clindamicina respectivamente en español).

Debido a su amplia documentación por tratarse de un desarrollo pionero en el ámbito universitario, el sistema Mycin ha servido de referencia constante en los trabajos de ingeniería del conocimiento y de representación del conocimiento, haciendo referencia a su arquitectura con representación en reglas y la forma de realizar razonamiento aproximado con factores de certeza. Se presenta aquí para ilustrar un caso de aplicación del método de clasificación heurística.

El problema que trata de resolver Mycin se centra principalmente en la identificación del organismo causante de una infección bacteriana (aunque en versiones posteriores incluyó también algunas infecciones víricas y producidas por hongos). Para ello, cuenta con observaciones correspondientes a dos tipos:

a) datos del organismo obtenidos en un laboratorio mediante la realización de un cultivo de una muestra obtenida del paciente (sangre, orina, tejidos, etc).

b) datos clínicos sobre el paciente respecto a su estado: dolor, inflamación, fiebre, hábitos, etc.

Los datos del organismo obtenidos en laboratorio mediante cultivo de muestras se basan en hacer crecer al organismo en condiciones favorables para su reproducción. Para facilitar su identificación se hacen observaciones al microscopio para conocer el tamaño, forma, crecimiento, etc., pruebas de tinción Gram o Ziehl Neelsen que pueden resultar positivas o negativas y pruebas de crecimiento en condiciones limitadas (por ejemplo con uso de ciertos antibióticos). El proceso de pruebas en laboratorio requiere 2 días de espera lo que permite normalmente una identificación completa del organismo.

Sin embargo, en ocasiones se desea suministrar algún tratamiento antibiótico desde el primer momento sin esperar los 2 días con el fin de limitar el crecimiento de la infección, aunque no se conozca con seguridad total la identidad del organismo. Para ello, Mycin intenta aprovechar la información disponible sobre el estado del paciente además de resultados parciales que se hayan podido adelantar del laboratorio para identificar el organismo que mejor corresponde a dichas observaciones. Otros factores que influyen en la necesidad de tratar el problema como una acumulación de indicios con certeza parcial que permita contrastar los datos del paciente con los datos del laboratorio son: (1) la extracción de la muestra debido a una manipulación incorrecta ha podido sufrir una contaminación externa y (2) en alguna ocasión el cultivo puede generar el desarrollo de bacterias que normalmente están presentes en pacientes sanos y que no generan estados patológicos.

¿Nombre del paciente?
> F.Braun
¿Sexo?
> Varón
¿Edad?
> 55
¿Hay cultivos para F.Braun relacionados con la presente enfermedad?
> Sí
Al primer cultivo se le denominará CULTIVO-1.
¿De qué lugar se extrajo el cultivo?
> Sangre
¿Fecha del cultivo?
> 20 junio 1977
Al organismo del CULTIVO-1 se le denominará ORGANISMO-1.
¿Cuál es la identidad de dicho organismo notificada por el laboratorio?
> Desconocido
¿Resultado de la tinción Gram del ORGANISMO-1?
> Negativo
¿Resultado de la tinción Ziehl-Neelsen del ORGANISMO-1?
> Desconocido
¿Forma del ORGANISMO-1 (bacilo, coco, etc.)?
> ¿por qué? *(el usuario pregunta por qué el sistema hace esta pregunta)*
... (explicación del sistema mostrando la regla que trata de aplicar)
> bacilo
..... *(siguen 12 preguntas más sobre características de organismos)* ....
¿Cuál es la supuesta puerta de entrada a la sangre de ORGANISMO-1?
> Tracto gastrointestinal (certeza 0.6)
¿Presenta quemaduras el paciente?
> No
¿El paciente es “huesped debilitado” (alcohólico, inmunodeprimido, etc.)?
> No
¿El paciente tiene fiebre?
> Sí
... (siguen unas 50 preguntas más) ...
Probables infecciones y organismos asociados:
INFECCION: Bacteriemia (presencia de bacterias en la sangre)
ORGANISMOS:
1. E. Coli
2. Klebsiella
3. Enterobacter
4. Klebsiella-pneumoniae
... (siguen preguntas relacionadas con la posible terapia) ...
La recomendación preferida es la siguiente:
Para cubrir a los organismos 1,2,3,4:
Suministrar: GENTAMICINA
Dosis: 119 mg cada 8 horas durante 10 días (calculado en base a 1.7 mg/kg)
Comentarios: modificar la dosis en caso de fallo renal.

Figura 3.25: Ejemplo de diálogo entre el sistema Mycin y el operador (traducción no literal de la versión en inglés original).

La figura 3.25 muestra un ejemplo de operación del sistema Mycin (ejemplo extraído de [Buchanan, Shortliffe, 84]). El sistema realiza una serie de preguntas sobre el paciente y datos de laboratorio sobre el cultivo a las que el usuario puede contestar con valores de certeza parcial o incluso indicar que no conoce la respuesta. A partir de esos datos, Mycin responde con los posibles gérmenes causantes de la infección. En una segunda parte (a la que correspondería otro tipo de método y que no se analiza en este apartado) Mycin genera una terapia recomendada en forma de antibióticos a suministrar para eliminar la infección. Además del tipo de conversación mostrado en el ejemplo, el sistema es capaz, durante la consulta y al final, de contestar a preguntas que justifican el razonamiento mostrando las reglas utilizadas.

La representación del conocimiento utilizada por el sistema original utilizaba ternas concepto-atributo-valor con reglas con medidas de incertidumbre (factores de certeza) con una única base de conocimiento en donde se reunía todo el conocimiento de una forma relativamente estructurada con reglas que incorporaban meta-conocimiento para determinar si los ciertos atributos eran conocidos. El lenguaje de programación era LISP lo que condicionó la sintaxis de los hechos y reglas. Por ejemplo, las siguientes sentencias son ejemplos de hechos (atributo-concepto-valor-certeza):

```
(IDENTITY ORGANISM-1 PSEUDOMONAS 0.8)
(SITE CULTURE-2 THROAT 1.0)
(BURNED PATIENT-298 YES -1.0)
```

Un ejemplo de regla con la representación original es la siguiente:

```
PREMISA:     ($AND (SAME CNTXT GRAM GRAMNEG)
                   (SAME CNTXT MORPH ROD)
                   (SAME CNTXT AIR ANAEROBIC))
ACCION:      (CONCLUDE CNTXT IDENTITY BACTERIOIDES .6)
```

Cuyo significado es el siguiente:

```
Si     1) el tipo del organismo es gram-negativo, y
       2) la morfología del organismo es bastón, y
       3) el organismo es anaeróbico
Entonces hay una evidencia de 0.6 sobre que la identidad del
organismo es bacterioide.
```

Desde el punto de la estructura de método de clasificación heurística se tienen los siguientes contenidos (figura 3.26):

1. Las *observaciones* son datos observados del paciente además de resultados sobre el cultivo correspondiente a la infección del paciente. Entre los datos del paciente se encuentran características generales tales como la edad, el sexo, etc. además de datos relacionados con el estado que presenta tales como niveles de leucocitos, glucosa, etc. en análisis de sangre, estado respecto a si presenta dolores, inflamación, quemaduras. Se tienen también dentro de este conjunto datos relacionados con la posible puerta de entrada del organismo (gastrointestinal, orina, etc.), si el paciente está hospitalizado, si ha sido tenido una intervención quirúrgica recientemente, etc. Por otra parte, los datos relacionados con las pruebas de laboratorio en cuanto a cultivo de la muestra están relacionados con el lugar de donde se extrajo la muestra (sangre, orina, tejido, etc.), el resultado de la tinción Gram (positiva o negativa), información derivada de la observación al microscopio tal como la morfología general (coco, bacilo), la forma de crecimiento (en cadenas, etc.), la forma específica (ojiva en cocos, fusiforme en bacilos, etc.).

2. Las *clases de observaciones* corresponden a información que se abstrae de las observaciones. Por ejemplo, la estimación de la superficie de la piel a partir del peso y estatura del paciente, abstracción del lugar de donde se

extrajo la muestra (por ejemplo, es de un lugar no estéril si se extrajo el tracto grastro-intestinal, de la piel, de las vías respiratorias, etc.), identificación del síntoma leucopenia que se deriva de que se tiene un nivel bajo de leucocitos (inferior al valor 2.500) o asociación del estado de inmunosupresión al paciente derivado de que presenta el síntoma leucopenia.

3. Las *clases de categorías* corresponden a clases generales de organismos bacterianos. Por ejemplo, el organismo es contaminante o no, el tipo de infección (bacteriemia en sangre, meningitis, endocarditis, etc.), la familia de la bacteria (estreptococo, enterobacteria, etc).

4. Las *categorías* identifican tipos específicos de organismos bacterianos. Por ejemplo: *Klebsiella-pneumoniae, Pseudomonas-aeruginosa, Bacilus-subtilis, Estreptococo-pneumoniae, Meningococo, Proteus-no-mirabilis, Hemofilus-influenzae, Difteroide, E. Coli, Salmonella* o *Serratia*.

La figura 3.27 muestra ejemplos de reglas de la base de conocimiento de Mycin que relacionan algunos de los términos anteriores. Por ejemplo la primera regla corresponde a un caso de abstracción (a partir de los datos del paciente sobre su peso y altura se estima la superficie de la piel). El resto de las reglas tienen en sus consecuentes categorías (generales o específicas) y corresponden a conocimiento de asociación heurística y refino.

La inferencia en Mycin se hacía por encadenamiento hacia atrás en la base de reglas. El sistema partía de una regla objetivo (la *goal rule*) que disparaba todo el proceso. En dicha regla la primera premisa hacía referencia a la existencia de un organismo que requiera terapia. El encadenamiento por dicha premisa hacia atrás llevaba a otras reglas a través de las cuales se alcanzaba el atributo *identidad* del organismo. Dicho atributo tenía como posibles valores todas las categorías posibles

de organismos bacterianos. La evaluación de las reglas que concluían sobre dicho atributo, encadenándose por otras reglas hacia las premisas que se preguntaban al usuario, daba lugar a las correspondientes conclusiones sobre identificación de las bacterias. Este enfoque resulta análogo al proceso descrito por el algoritmo de clasificación heurística dirigida por los objetivos.

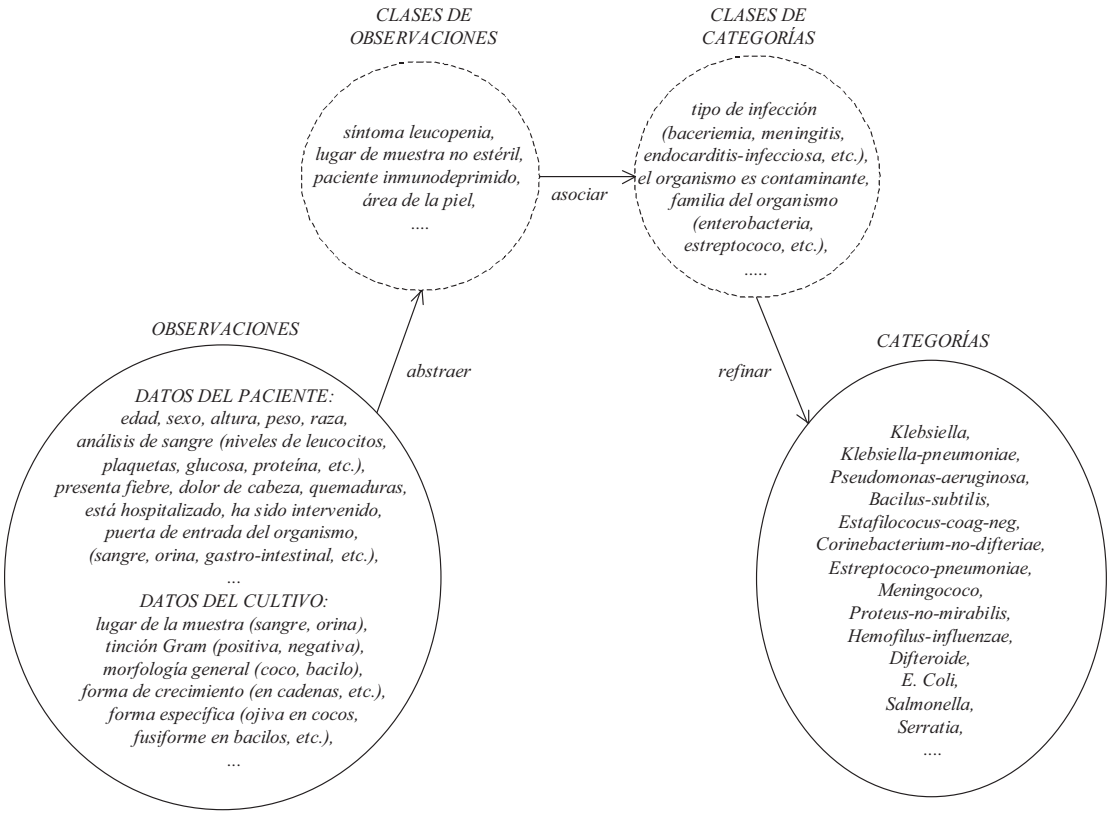


Figura 3.26: Ejemplos de contenidos en el problema de identificación de bacterias en los diferentes espacios considerados por el método de clasificación heurística.

```
SI:     1) el peso del paciente es P, y
        2) la altura del paciente es H
ENTONCES:
calcular el área de la superficie de la piel por el algoritmo Boyd con P y H.

SI      1) el lugar de la muestra es la sangre, y
        2) hay como máximo un cultivo positivo para esta muestra, y
        3) el paciente tiene fiebre de origen desconocido,
        4) el paciente ha tenido cirugía cardiaca reciente,
ENTONCES Hay una evidencia de 0.8 de tipo de infección endocarditis-infecciosa

SI      1) el lugar de la muestra es estéril, y
        2) el paciente no es huésped comprometido, y
        3) el paciente tiene fiebre, y
        4) A - el recuento de leucocitos es mayor de 10.5, o
           B - el porcentaje de PMNs es mayor que 78, o
           C - el porcentaje de leucocitos con bandas es mayor que 10
ENTONCES Hay una evidencia de 0.8 de que el organismo no es contaminante

SI      1) el tipo de organismo es gram-negativo, y
        2) la morfología del organismo es bacilo, y
        3) el organismo es aeróbico
ENTONCES: Hay una evidencia de 0.8 de que el organismo es enterobacteria

SI      1) el tipo de infección es bacteriemia primaria, y
        2) el lugar de la muestra es estéril, y
        3) la puerta de entrada del organismo es gastrointestinal
ENTONCES: Hay una evidencia de 0.8 de que el organismo es enterobacteria

SI:     1) el tipo de organismo es gram-negativo, y
        2) la morfología del organismo es bacilo, y
        3) la puerta de entrada del organismo es por orina, y
        4) el lugar del cultivo es la sangre, y
        5) el paciente no ha tenido tratamiento urinario, y
        6) el paciente no ha sido tratado por cistitis
ENTONCES Hay una evidencia de 0.6 de que la identidad del organismo es E.Coli

SI      1) el tipo de organismo es gram-positivo, y
        2) la morfología del organismo es coco, y
        3) la forma de crecimiento es en cadenas
ENTONCES: Hay una evidencia de 0.7 de que el organismo es Estreptococo
```

Figura 3.27: Ejemplos de reglas de la base de conocimiento de Mycin.

El proceso después continuaba desarrollando otra búsqueda dirigida a generar terapias mediante suministro de antibióticos para eliminar las infecciones producidas por los gérmenes identificados. Este proceso lo iniciaba la *goal rule* y se realizaba una búsqueda sencilla similar a un proceso de generación-y-prueba de la siguiente forma de acuerdo con la versión debida a Clancey que mejoró la versión inicial [Clancey, 84]. Para cada organismo se elegía un número N fijo de antibióticos candidatos (por ejemplo, N = 3) ordenados por preferencia atendiendo a criterios individuales de cada organismo. A continuación se iniciaba un proceso de generación de combinaciones y prueba orientado a que las combinaciones de antibióticos fueran aceptables atendiendo a criterios globales. Este proceso progresivamente elegía antibióticos de la lista ordenada de los organismos. Dicha generación se verificaba después con el fin comprobar que (1) se cubrían todos los organismos, (2) no había antibióticos redundantes y (3) no se violaba alguna contraindicación del paciente. Este algoritmo, que proporcionaba un rendimiento aceptable según sus autores, corresponde a una parte del sistema Mycin que resuelve un problema que no es de clasificación. En este caso, cada elemento del espacio de soluciones es una combinación de antibióticos por lo que dicho espacio no es un conjunto de categorías prefijado sino que se genera conforme se resuelve el problema.

3.4.2. Ejemplo 2: Diagnóstico de enfermedades hepáticas

Para ilustrar el proceso de razonamiento del método de clasificación jerárquica considérese el siguiente ejemplo inspirado en el sistema MDX [Mittal et al., 79; Chandrasekaran, Mittal, 83] cuyo fin era diagnosticar enfermedades del hígado. El sistema contaba con una jerarquía de hipótesis de enfermedades que dirigía el proceso de búsqueda de acuerdo con un enfoque de clasificación jerárquica y se verificaba mediante el proceso de establecer y refinar.

En este ejemplo se tiene una jerarquía de enfermedades del hígado de forma que a partir de la raíz general se tienen tres opciones hepatitis, colestasis y cirrosis. Una forma de confirmar hepatitis es cuando se tiene alguna transaminasa alta (por ejemplo la transaminasa GOT o la transaminasa GPT) y, además, se tiene ictericia, es decir, bilirrubina alta en la sangre. A su vez hepatitis tiene tres opciones: A, B y C. Por otra parte, la opción colestasis consiste en una supresión o bloqueo de la secreción de la bilis al duodeno.

La presencia de niveles altos de bilirrubina es debido a una retención de bilis. Esto puede ser debido a un bloqueo del conducto debido a una piedra en vesícula biliar, una presión externa (por ejemplo por un tumor en órganos cercanos) o por colangitis, es decir, una inflamación del conducto. La piedra en vesícula biliar se suele detectar mediante ecografía abdominal. La piedra además causa dolor abdominal debido a la obstrucción e incremento de movimientos peristálticos del conducto.

La jerarquía de hipótesis de enfermedades se representa mediante un conjunto de implicaciones de la siguiente forma:

```
J1: enfermedad(paciente)=enfermedad-higado
    → enfermedad(paciente)=hepatitis o
       enfermedad(paciente)=colestasis o
       enfermedad(paciente)=cirrosis

J2: enfermedad(paciente)=hepatitis
    → enfermedad(paciente)=hepatitis-A o
       enfermedad(paciente)=hepatitis-B o
       enfermedad(paciente)=hepatitis-C

J3: enfermedad(paciente)=colestasis
    → enfermedad(paciente)=piedra-vesícula-biliar o
       enfermedad(paciente)=tumor-zona-abdominal o
       enfermedad(paciente)=colangitis

J4: enfermedad(paciente)=tumor
    → enfermedad(paciente)=tumor-conducto o
       enfermedad(paciente)=tumor-vesícula-biliar o
       enfermedad(paciente)=tumor-páncreas
```

La base de conocimiento (parcial) de relaciones heurísticas es:

```
H1: coluria(paciente)=sí
    → enfermedad(paciente)=enfermedad-hígado

H2: nivel(transaminasas)=alto y
    ictericia(paciente)=sí
    → enfermedad(paciente)=hepatitis

H3: nivel(transaminasas)= normal
    → NO(enfermedad(paciente)=hepatitis)

H4: ictericia(paciente)=sí
    → secreción(bilis)=retenida

H5: secreción(bilis)=retenida
    → enfermedad(paciente)=colestasis

H6: secreción(bilis)=normal
    → NO(enfermedad(paciente)=colestasis)

H7: resultado(ecografía-abdominal)=positivo
    → enfermedad(paciente)=piedra-vesícula-biliar

H8: resultado(ecografía-abdominal)=normal
    → NO(enfermedad(paciente)=piedra-vesícula-biliar)
```

La base de conocimiento (parcial) de criterios de abstracción es:

```
A1:  coloración(orina)=oscura → coluria(paciente)=sí

A2:  coloración(orina)=normal → coluria(paciente)=no

A3:  medida(bilirrubina) =< 0.3 → nivel(bilirrubina)=normal

A4:  medida(bilirrubina) > 0.3 → nivel(bilirrubina)=alto

A5:  medida(GPT) =< 35 → nivel(GPT)=normal

A6:  medida(GPT) > 35 → nivel(GPT)=alto

A7:  nivel(bilirrubina)=alto → ictericia(paciente)=sí

A8:  coloración(piel)=amarilla → ictericia(paciente)=sí

A9:  nivel(bilirrubina)=normal → ictericia(paciente)=no

A10: nivel(GPT)=alto → nivel(transaminasas)=alto

A11: nivel(GPT)=normal → nivel(transaminasas)=normal
```

El procedimiento de inferencia de clasificación jerárquica parte la hipótesis más general (enfermedad del hígado). A continuación se muestra el orden de llamadas a los pasos de inferencia indicando los cambios que se producen en los conjuntos considerados en el algoritmo:

```
hipótesis = 'enfermedad(paciente)=enfermedad-hígado'

observaciones = {}

1.  probar(hipótesis, observaciones -> prueba, observaciones-requeridas)
    Reglas que concluyen sobre enfermedad(paciente)=enfermedad-hígado:
    H1: coluria(paciente)=sí → enfermedad(paciente)=enfermedad-hígado
    prueba = DESCONOCIDO
    observaciones-requeridas = {coluria(paciente)}
2.  adquirir(observaciones-requeridas, observaciones -> observaciones)ç
    Reglas que concluyen sobre coluria(paciente):
    A1: coloración(orina)=oscura → coluria(paciente)=sí
    A2: coloración(orina)=normal → coluria(paciente)=no
    PREGUNTA-SISTEMA: ¿coloración(orina)?
    RESPUESTA-USUARIO: oscura
    observaciones = {coloración(orina)=oscura, coluria(paciente)=sí}
3.  probar(hipótesis, observaciones -> prueba, observaciones-requeridas)
    prueba = VERDADERO
4.  especificar(hipótesis -> conjunto-hipótesis)
    Regla con 'enfermedad(paciente)=enfermedad-hígado' de antecedente:
    J1:enfermedad(paciente)=enfermedad-higado
       → enfermedad(paciente)=hepatitis o
         enfermedad(paciente)=colestasis o
         enfermedad(paciente)=cirrosis
    conjunto-hipótesis = {enfermedad(paciente)=hepatitis,
                          enfermedad(paciente)=colestasis,
```

enfermedad(paciente)=cirrosis}

hipótesis = 'enfermedad(paciente)=hepatitis'

5. probar(hipótesis, observaciones -> prueba, observaciones-requeridas)

Reglas que concluyen sobre enfermedad(paciente)=hepatitis:

H2: nivel(transaminasas)=alto y ictericia(paciente)=sí

→ enfermedad(paciente)=hepatitis

H3: nivel(transaminasas)=normal → NO(enfermedad(paciente)=hepatitis)

prueba = DESCONOCIDO

observaciones-requeridas = {nivel(transaminasas)}

6. adquirir(observaciones-requeridas, observaciones -> observaciones)

Reglas que concluyen sobre nivel(transaminasas):

A10: nivel(GPT)=alto → nivel(transaminasas)=alto

A11: nivel(GPT)=normal → nivel(transaminasas)=normal

Reglas que concluyen sobre nivel(GPT):

A5: medida(GPT) =< 35 → nivel(GPT)=normal

A6: medida(GPT) > 35 → nivel(GPT)=alto

PREGUNTA-SISTEMA: ¿medida(GPT)?

RESPUESTA-USUARIO: 31

observaciones = {medida(GPT)=31, nivel(GPT)=normal, nivel(transaminasas)=normal,

coloración(orina)=oscura, coluria(paciente)=sí}

7. probar(hipótesis, observaciones -> prueba, observaciones-requeridas)

prueba = FALSO

se elige el siguiente valor de hipótesis (ver paso 4):

hipótesis = 'enfermedad(paciente)=colestasis'

8. probar(hipótesis, observaciones -> prueba, observaciones-requeridas)

Reglas que concluyen sobre enfermedad(paciente)=colestasis

H5: secreción(bilis)=retenida → enfermedad(paciente)=colestasis

H6: secreción(bilis)=normal → NO(enfermedad(paciente)=colestasis)

Reglas que concluyen sobre secreción(bilis):

H4: ictericia(paciente)=sí → secreción(bilis)=retenida

```
    prueba = DESCONOCIDO
    observaciones-requeridas = {ictericia(paciente)}
9.  adquirir(observaciones-requeridas, observaciones -> observaciones)
    Reglas que concluyen sobre ictericia(paciente):
    A7:  nivel(bilirrubina)=alto → ictericia(paciente)=sí
    A8:  coloración(piel)=amarilla → ictericia(paciente)=sí
    A9:  nivel(bilirrubina)=normal → ictericia(paciente)=no
    Reglas que concluyen sobre nivel(bilirrubina):
    A3:  medida(bilirrubina) =< 0.3 → nivel(bilirrubina)=normal
    A4:  medida(bilirrubina) > 0.3 → nivel(bilirrubina)=alto
    PREGUNTA-SISTEMA: ¿medida(bilirrubina)?
    RESPUESTA-USUARIO: 1.8
    observaciones = {medida(bilirrubina)=1.8, nivel(bilirrubina)=alto,
                     paciente(ictericia)=sí, medida(GPT)=31, nivel(GPT)=normal,
                     nivel(transaminasas)=normal, coloración(orina)=oscura,
                     coluria(paciente)=sí}
10. probar(hipótesis, observaciones -> prueba, observaciones-requeridas)
    prueba = VERDADERO
11. especificar(hipótesis -> conjunto-hipótesis)
    Regla con 'enfermedad(paciente)=colestasis' de antecedente:
    J2:enfermedad(paciente)=colestasis
       → enfermedad(paciente)= piedra-vesícula-biliar o
         enfermedad(paciente)= tumor-zona-abdominal o
         enfermedad(paciente)= colangitis
    conjunto-hipótesis = {enfermedad(paciente)=piedra-vesícula-biliar,
                          enfermedad(paciente)=tumor-zona-abdominal,
                          enfermedad(paciente)=colangitis}
    hipótesis = 'enfermedad(paciente)=piedra-vesícula-biliar'
12. probar(hipótesis, observaciones -> prueba, observaciones-requeridas)
    Reglas que concluyen sobre enfermedad(paciente)=piedra-vesícula-biliar:
```

```
    H7: resultado(ecografía-abdominal)=positivo
        → enfermedad(paciente)=piedra-vesícula-biliar
    H8: resultado(ecografía-abdominal)=normal
        → NO(enfermedad(paciente)=piedra-vesícula-biliar)
     prueba = DESCONOCIDO
     observaciones-requeridas = {resultado(ecografía-abdominal)}
13. adquirir(observaciones-requeridas, observaciones -> observaciones)
    No hay reglas que concluyen sobre resultado(ecografía-abdominal)
     PREGUNTA-SISTEMA: ¿resultado(ecografía-abdominal)?
     RESPUESTA-USUARIO: positivo
     observaciones = {resultado(ecografía-abdominal)=positivo,
                      medida(bilirrubina)=1.8, nivel(bilirrubina)=alto,
                      paciente(ictericia)=sí, medida(GPT)=31, nivel(GPT)=normal,
                      nivel(transaminasas)=normal, coloración(orina)=oscura,
                      coluria(paciente)=sí}
14. probar(hipótesis, observaciones -> prueba, observaciones-requeridas)
     prueba = VERDADERO
15. especificar(hipótesis -> conjunto-hipótesis)
    conjunto-hipótesis = {}
    soluciones = {enfermedad(paciente)=piedra-vesícula-biliar}
....... el procedimiento puede continuar buscando otras soluciones
```

La primera solución encontrada es:

```
    enfermedad(paciente)=piedra-vesícula-biliar
```

El proceso de interacción con el usuario se adapta a las respuestas que se van dando, de forma que las últimas preguntas son consecuencia de las respuestas de las anteriores. La secuencia es:

```
PREGUNTA-SISTEMA: ¿coloración(orina)?

RESPUESTA-USUARIO: oscura

PREGUNTA-SISTEMA: ¿medida(GPT)?

RESPUESTA-USUARIO: 31

PREGUNTA-SISTEMA: ¿medida(bilirrubina)?

RESPUESTA-USUARIO: 1.8

PREGUNTA-SISTEMA: ¿resultado(radiografía-abdominal)?

RESPUESTA-USUARIO: positivo
```

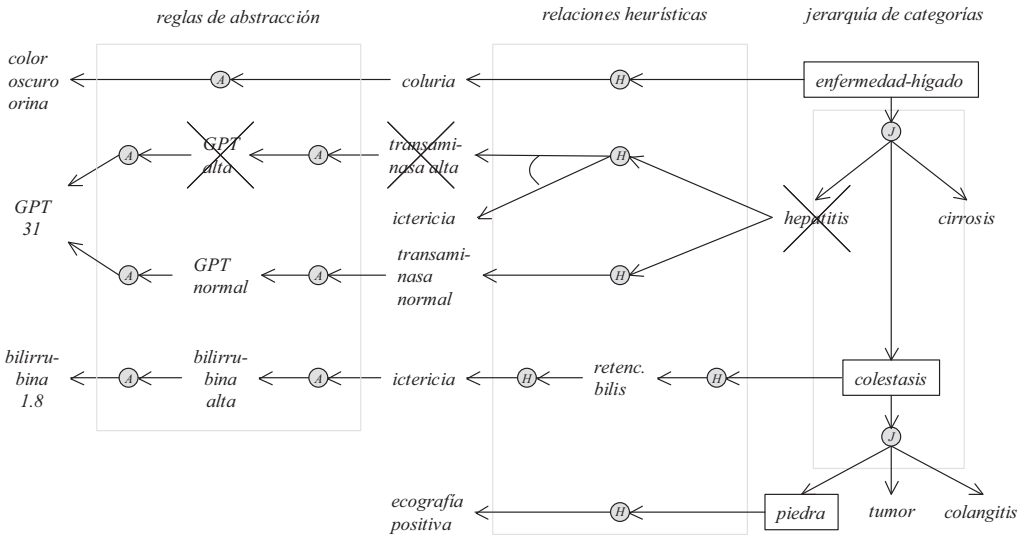


Figura 3.28: Gráfico resumen de la búsqueda realizada (los nodos con letra H representan relaciones heurísticas, los de letra A representan reglas de abstracción y los J sobre jerarquía de soluciones).

Este ejemplo muestra un caso muy simplificado para facilitar la claridad de la descripción. En casos reales las bases de conocimiento tienen una dimensión y complejidad mayor y las reglas pueden incluir valores de incertidumbre que se acumulan sobre las hipótesis consideradas. Como alternativa a la representación en reglas, la jerarquía de categorías puede integrarse con el conocimiento de relaciones heurísticas mediante una representación con marcos, en donde cada marco es un patrón de enfermedad y los *slots* incluyen los síntomas o clases de síntomas que tiene dicha enfermedad. El procedimiento de control de la equiparación establece la forma en que se confirma cada hipótesis a partir de los síntomas dando, por ejemplo, más peso a algunos síntomas que a otros. Aquí también cabe la utilización de procedimientos de tratamiento de la incertidumbre.

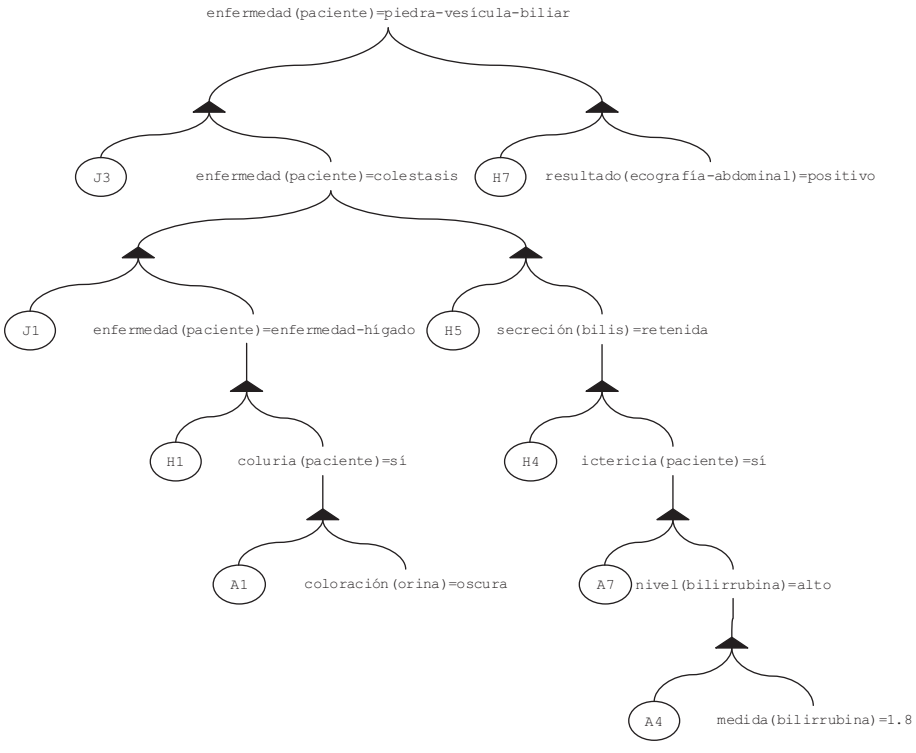


Figura 3.29: Árbol explicativo de cómo se ha obtenido cada conclusión.

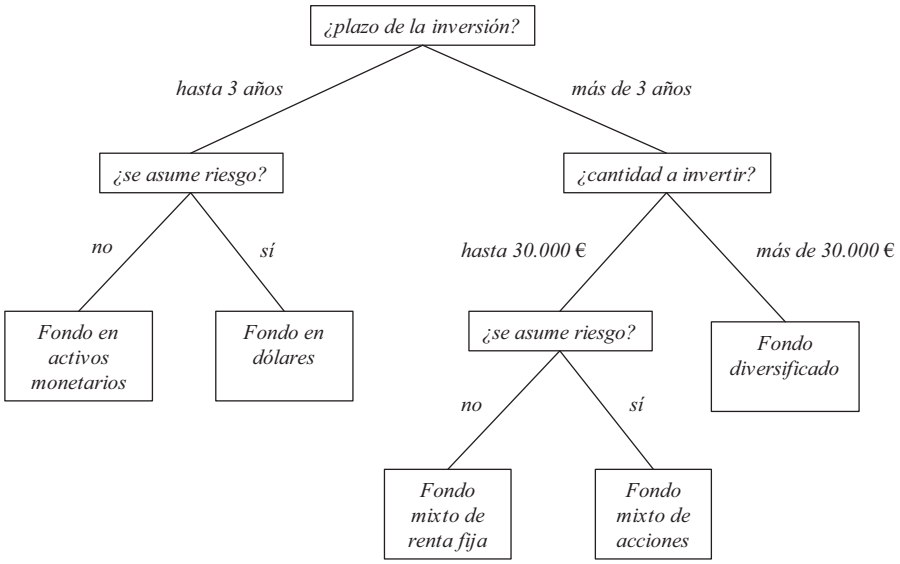


Figura 3.30: Ejemplo de árbol de decisión.

3.4.3. Ejemplo 3: Recomendación de tipo de inversión

Un árbol de decisión es una estructura de representación ampliamente utilizada en diferentes disciplinas. En un árbol de decisión cada nodo del árbol es una pregunta y cada arco indica una de las posibles respuestas. Las hojas del árbol representan las soluciones a las que se llega tras contestar a las diversas preguntas, partiendo desde la raíz. Por ejemplo, en la figura 3.30 se muestra un árbol de decisión para decidir un fondo de inversión. Partiendo de la raíz si, por ejemplo, se contesta con que el plazo de inversión es mayor de 3 años se pasaría a la pregunta sobre cantidad a invertir. Si a continuación se contesta hasta 30.000€ se pasaría a la pregunta de riesgo asumido. Si en ese punto se contesta con la respuesta no, se alcanzaría finalmente como resultado fondo mixto de renta fija.

Muchos ámbitos profesionales cuentan con árboles de decisión para expresar condiciones que se deben dar para alcanzar las diferentes conclusiones. Por ejemplo, además de recomendaciones sencillas sobre posibilidades de inversión económica, pueden expresarse criterios de clasificación en zoología o botánica, decisiones simples sobre productos a comprar, etc. Es frecuente encontrar documentación profesional expresada de esta forma y es conveniente conocer las semejanzas y diferencias que presenta con respecto a otros métodos. En particular se presenta aquí la relación de este tipo de estructura con respecto al método de clasificación jerárquica, con objeto de resaltar las capacidades de representación que tiene dicho método.

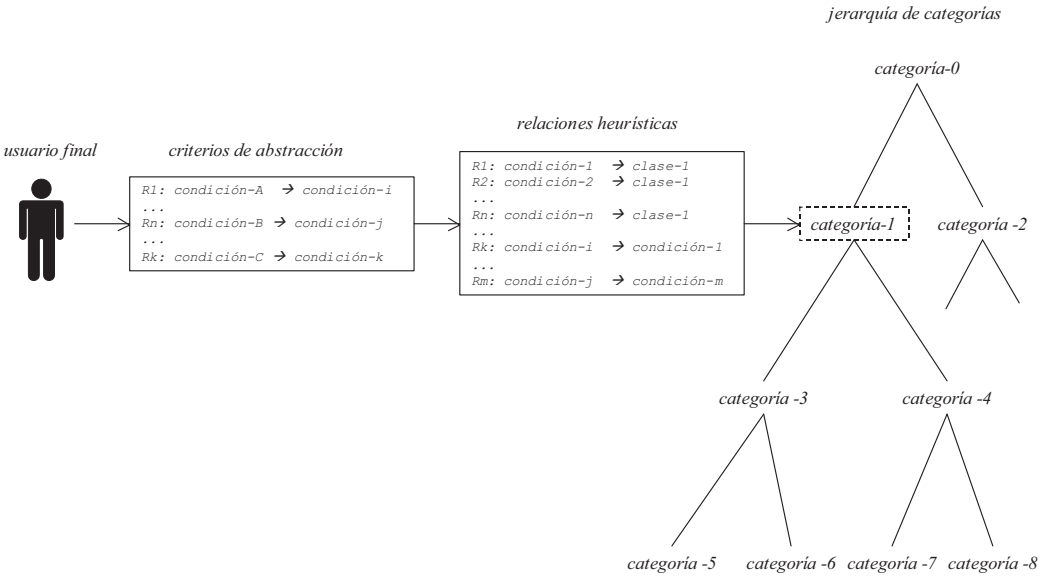


Figura 3.31: Visión general del método de clasificación jerárquica.

Por su parte, el método de clasificación jerárquica maneja otro tipo de jerarquía, la jerarquía de categorías, en donde cada nodo es una categoría no una pregunta como en el árbol de decisión (figura 3.31). La búsqueda hacia la solución se realiza haciendo un recorrido desde la raíz, al igual que en el árbol de decisión. En cada momento, se plantea una hipótesis correspondiente al nodo en curso (por ejemplo, en la figura 3.31 el nodo denominado categoría-1) y se analiza el conocimiento que

permite confirmar o descartar dicho nodo. Para ello, si se manejan reglas como representación, se analizan las reglas que concluyen sobre esa hipótesis, encadenándose hacia atrás por el conocimiento de relaciones heurísticas y, después, el conocimiento de abstracción hasta alcanzar observaciones que, en su caso, deben preguntarse al usuario final. Debe tenerse en cuenta que la confirmación de una hipótesis en general dispondrá de varias alternativas, tantas como los diversos caminos que forman el encadenamiento en las reglas. De esta forma, si el usuario final no es capaz de contestar a una de las preguntas que le hace el sistema por desconocer la respuesta, el sistema tendrá la posibilidad de utilizar otra alternativa asociada a otras preguntas que permitan confirmar o descartar la hipótesis.

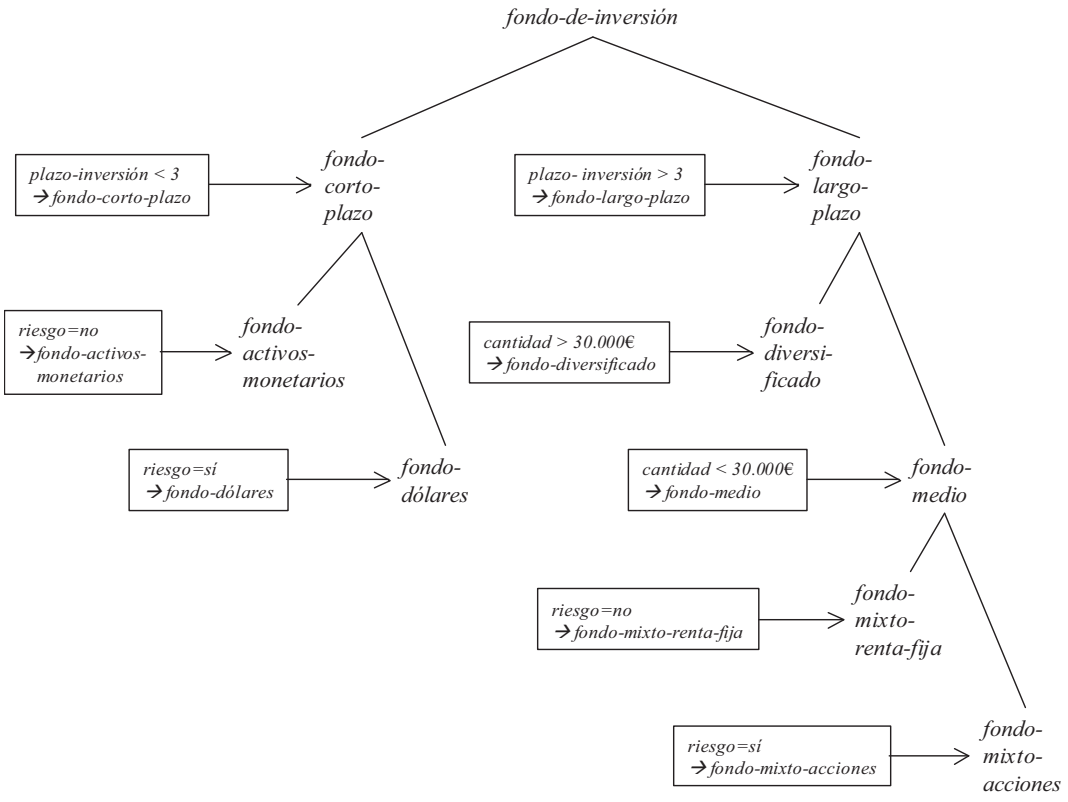


Figura 3.32: Modelo para clasificación jerárquica correspondiente al árbol de decisión.

La figura 3.32 muestra la construcción de un modelo para dar soporte al método de clasificación jerárquica, desarrollado únicamente a partir del conocimiento existente del árbol de la decisión de la figura 3.30. Se maneja una jerarquía con tantos nodos solución hoja como se tiene en el árbol de decisión. Para cada respuesta de una pregunta hay un nodo categoría. Para confirmar cada nodo hay una única regla, cuya parte de condición incluye la pregunta con la respuesta del árbol de decisión correspondiente a dicho nodo.

En dicho ejemplo se observa que, para confirmar una hipótesis de categoría, únicamente tiene una posibilidad (indicada por la regla asociada) de forma que, en caso de que el usuario no pueda responder, el proceso se detiene ahí sin poder continuar con soluciones alternativas tal como es posible en un modelo más completo en clasificación jerárquica. Por otro lado, las preguntas que se plantean al usuario son directamente las que se consideran para alcanzar la hipótesis que, en muchas ocasiones, pueden resultar demasiado técnicas y susceptibles de ser traducidas a un lenguaje más cercano del usuario final, tal como realiza el método de clasificación jerárquica con ayuda del conocimiento de abstracción.

En consecuencia, los árboles de decisión deben contemplarse como una estructura de representación sencilla, intuitiva que puede utilizarse como fuente parcial de información para adquisición del conocimiento, pero que presenta limitaciones para ser utilizados en problemas complejos en donde el método de clasificación jerárquica puede ofrecer mayores posibilidades.

3.5 Ejercicios

EJERCICIO 3.1. Considérese el siguiente conjunto de sentencias relativas al problema de recomendación del deporte más adecuado para una persona de acuerdo con sus preferencias y condiciones físicas:

```
S1:  Si deporte-aeróbico y preferencia-aire-libre entonces ciclismo
S2:  Si lesión-en-rodilla entonces estado-fisico-disminuído
S3:  Si deporte-anaeróbico y forma-física-mala entonces musculación
S4:  Si problemas-cardíacos entonces estado-físico-disminuído
S5:  Si peso entre 60 y 80 Kg y altura entre 1.60 y 1.70 entonces
     complexión-media
S6:  Si forma-física-aceptable y objetivo-perder-peso entonces
     deporte-aeróbico
S7:  Si complexión-fuerte y deportista-experimentado entonces
     capacidad-deportiva-alta
S8:  Si años-practicando-deporte < 1 entonces deportista-no-
     experimentado
```

SE PIDE:

a) Teniendo en cuenta que desea aplicar el método de *clasificación heurística*, describir con ejemplos cuál es el espacio de observaciones y el espacio de categorías correspondiente a este problema.
b) Responder a qué clase de conocimiento corresponde cada una de las sentencias anteriores de acuerdo con los tipos de conocimiento del método de *clasificación heurística*.
c) ¿Qué estrategia de razonamiento (tipo de algoritmo) de los posibles a utilizar en clasificación heurística es apropiada para ser aplicada en este problema?.

EJERCICIO 3.2. Considérese un modelo diagnóstico de fallos del sistema eléctrico de un automóvil. Dicho modelo está formulado de acuerdo con el método de *clasificación heurística*. Los hechos que se manejan son los siguientes:

Concepto	**Atributo**	**Valores**	**Descripción**
faros	iluminan	{sí, no}	Indica si encienden los faros
luz-interior	ilumina	{sí, no}	Indica si enciende la luz interior
claxon	sonido	{sí, no}	Indica si suena el claxon
limpiaparabrisas	responde	{sí, no}	Indica si responde el limpiaparabrisas
elevalunas	responde	{sí, no}	Indica si responde el elevalunas
vehículo	arranque	{sí, no, con-dificultad}	Indica cómo arranca el vehículo
	meses-detenido	número	Meses que ha estado detenido
	detención	{corta, prolongada}	Tipo de detención del vehículo
	respuesta-eléctrica	{normal, fallo-generalizado, fallo-aislado}	Respuesta de los elementos eléctricos del vehículo
iluminación	intensidad	{normal, baja, nula}	Intensidad de la iluminación
	estado	{normal, fallo}	Estado de la iluminación
operación	estado	{normal, fallo}	Estado de la operación
sistema-eléctrico	intensidad	{normal, baja, nula}	Intensidad en el sistema eléctrico
	estado	{normal, averiado}	Estado del sistema eléctrico
batería	antigüedad	{baja, alta}	Nivel de antigüedad de la batería
	años	número	Años de antigüedad de la batería
	estado	{normal, descargada}	Estado de la batería
motor-eléctrico	estado	{normal, averiado}	Estado del motor eléctrico
motor-arranque	estado	{normal, averiado}	Estado del motor de arranque
motor-elevalunas	estado	{normal, averiado}	Estado del motor del elevalunas
motor-limpiaparabrisas	estado	{normal, averiado}	Estado del motor del limpiaparabrisas
sistema-alimentación	estado	{normal, averiado}	Estado del sistema de alimentación
fusible	estado	{normal, fundido}	Estado de un fusible

De acuerdo con el método de *clasificación heurística* se dispone del siguiente *conocimiento de abstracción* expresado en forma de reglas:

```
A1: iluminan(faros)=no o ilumina(luz-interior)=no
    → estado(iluminación)=fallo

A2: sonido(claxon)=no o responde(limpiaparabrisas)=no o
    responde(elevalunas)=no
    → estado(operación)=fallo

A3: estado(iluminación)=fallo y estado(operación)=fallo
    → respuesta-eléctrica(vehículo)=fallo-generalizado

A4: arranque(vehículo)=con-dificultad
    intensidad(iluminación)=baja
    → intensidad(sistema-eléctrico)=baja

A5: meses-detenido(vehículo) > 6
    → detención(vehículo)=prolongada

A6: años(batería) > 5
    → antigüedad(batería)=alta
```

Las *relaciones heurísticas* correspondientes a este modelo se expresan con las siguientes reglas:

```
H1: estado(iluminación)=fallo o estado(operación)=fallo o
    intensidad(sistema-eléctrico)=baja
    → estado(sistema-eléctrico)=averiado

H2: sonido(claxon)=sí y estado(operación)=fallo
    → estado(motor-eléctrico)=averiado

H3: estado(iluminación)=fallo o
    intensidad(sistema-eléctrico)=baja
    → estado(sistema-alimentación)=averiado

H4: respuesta-eléctrica(vehículo)=fallo-generalizado
    → estado(batería)=descargada
```

```
H5: estado(iluminación)=fallo y detención(vehículo)=prolongada
    → estado(batería)=descargada

H6: detención(vehículo)=prolongada y antigüedad(batería)=alta
    → estado(batería)=descargada

H7: intensidad(sistema-eléctrico)=baja y
    antigüedad(batería)=alta
    → estado(batería)=descargada

H8: estado(iluminación)=fallo y sonido(claxon)=sí
    → estado(fusible)=fundido

H9: estado(operación)=fallo y estado(iluminación)=normal
    → estado(fusible)=fundido
```

Por último, el conocimiento sobre *jerarquía de categorías* está expresado mediante las siguientes sentencias (se asume que las hipótesis del mismo nivel son excluyentes):

```
J1: estado(sistema-eléctrico)=averiado
    → estado(motor-eléctrico)=averiado o
      estado(sistema-alimentación)=averiado

J2: estado(motor-eléctrico)=averiado
    → estado(motor-arranque)=averiado o
      estado(motor-elevalunas)=averiado o
      estado(motor-limpiaparabrisas)=averiado

J3: estado(sistema-alimentación)=averiado
    → estado(batería)=descargada o
      estado(fusible)=fundido
```

SE PIDE:

1) Aplicar la estrategia de inferencia de *clasificación jerárquica* (*establecer y refinar*) para determinar cuál puede ser la posible avería sabiendo únicamente que los faros no encienden, el claxon suena, el limpiaparabrisas funciona y que la batería tiene 6 años de antigüedad.

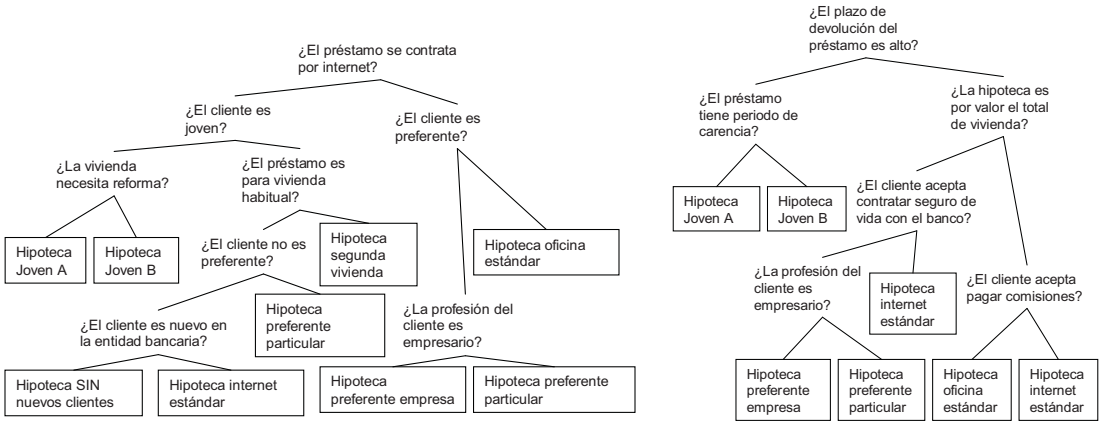
2) Explicar si la estrategia de inferencia de *clasificación heurística dirigida por los datos* sería aplicable en este modelo y qué ventajas o inconvenientes traería consigo. Mostrar cómo operaría el método con los datos considerados en el apartado (1).

3) Razonar igual que en (2) para el caso de la estrategia de inferencia de *clasificación heurística dirigida por los objetivos*.

EJERCICIO 3.3. Una determinada entidad bancaria ofrece diferentes tipos de préstamos hipotecarios destinados a la compra de una vivienda. Según las características de cada cliente que solicita un préstamo, la entidad selecciona el producto hipotecario más adecuado. Los tipos de préstamos hipotecarios que ofrece la entidad son los siguientes:

Tipo de préstamo hipotecario
Hipoteca joven A
Hipoteca joven B
Hipoteca SIN nuevos clientes
Hipoteca internet estándar
Hipoteca segunda vivienda
Hipoteca preferente particular
Hipoteca preferente empresa
Hipoteca oficina estándar

Algunos de los criterios que utiliza la entidad bancaria para elegir el préstamo más adecuado se pueden describir en forma de árboles de decisión. En estos árboles, la rama de la izquierda debajo de cada nodo corresponde a la respuesta afirmativa de la pregunta que representa el nodo (y la rama de la derecha es la respuesta negativa). Las hojas de los árboles son las elecciones que se realizan.



Otros criterios que utiliza la entidad bancaria para conceder y seleccionar el préstamo hipotecario son los siguientes:

- La cantidad mínima de préstamo a solicitar es de 50.000 euros y la máxima de 1.000.000 de euros.
- La cantidad solicitada para préstamo no debe superar el valor de la vivienda que se desea comprar.
- Para conceder el préstamo hipotecario, los ingresos mensuales del cliente deben superar los 2.000 euros excepto para la hipoteca joven (A o B) que deben superar los 1.000 euros.
- El cliente se considera joven si su edad es menor de 35 años. El cliente se considera nuevo en la entidad bancaria si su antigüedad no es superior a 2 años
- Se considera cliente preferente (a) si es empresario y tiene un flujo de caja superior a 500.000 euros o (b) si es empleado por cuenta ajena, tiene la nómina domiciliada y su antigüedad en el banco es de más de 10 años.

- El plazo de devolución del préstamo hipotecario debe estar entre 10 y 40 años. Más de de 25 años se considera un plazo alto.
- Se dice que el préstamo tiene 'periodo de carencia' si el primer año el cliente no desea pagar ninguna cuota mensual.

Se desea representar el problema del préstamo hipotecario utilizando el método *clasifiación jerárquica.*

SE PIDE:

1.1) Vocabulario conceptual descrito en forma de conceptos, atributos y los valores posibles de cada atributo.

1.2) Base de conocimiento de criterios abstracción, indicando para cada criterio de qué tipo se trata (agregación cuantitativa, interpretación cualitativa, generalización o de definición).

1.3) Base de conocimiento de jerarquía de categorías.

1.4) Base de conocimiento de asociaciones heurísticas.

Aplicar el proceso de inferencia del método de clasificación jerárquica para encontrar el tipo de préstamo más adecuado para un caso de un cliente que solicita un préstamo de 100.000 euros para la compra de una vivienda valorada en 200.000 euros cuya contratación se realiza por internet. En caso de que durante el proceso de resolución del problema sea necesario conocer más datos considerar los siguientes valores: plazo de devolución del préstamo 20 años, edad del cliente 38, es empleado por cuenta ajena, sus ingresos mensuales son 2.300 euros, el préstamo se destina a la compra de una vivienda habitual, el cliente tiene una antigüedad en el banco de 12 años, tiene nómina domiciliada y no acepta pago de comisiones.

SE PIDE:

2.1) Mostrar la lista de preguntas del sistema y las respuestas del usuario en el orden en que se realizan durante la inferencia Indicar cuál es la solución o soluciones encontradas.

2.2) Dibujar el gráfico resumen de la búsqueda realizada durante el proceso de inferencia.

2.3) Mostrar el proceso de resolución del problema indicando las sucesivas llamadas a los pasos de inferencia. Es suficiente con mostrar los primeros pasos de inferencia.

2.4) Indicar ventajas (si existen) del uso del método de clasificación heurística en este problema frente al uso de una representación en forma de base de datos (por ejemplo, con modelo relacional) con lenguajes de consulta para buscar préstamos hipotecarios a partir de sus características.

2.5) Indicar ventajas (si existen) del uso del método de clasificación heurística en este problema frente al uso de una representación en forma de árbol de decisión para seleccionar préstamos hipotecarios a partir de sus características.

4 Diagnóstico basado en modelos

El problema de diagnóstico puede contemplarse como un caso particular del problema de clasificación y ha recibido especial atención en dominios diversos tales como diagnóstico de sistemas eléctricos, plantas químicas, redes de distribución de agua, centrales térmicas, circuitos digitales, etc. En estos dominios es frecuente disponer de conocimiento detallado sobre el sistema a diagnosticar por lo que es posible contar con un modelo del funcionamiento de dicho sistema que sirva como base al proceso de búsqueda de fallos.

En este capítulo se describe un método orientado a problemas de diagnóstico que, a diferencia del caso de clasificación heurística, hace uso de dicho modelo de funcionamiento del sistema a diagnosticar. En el capítulo se estudian dos aproximaciones diferentes en función del tipo de modelo a considerar.

4.1 Descripción general

El proceso de diagnóstico utiliza los siguientes términos:

- *comportamiento:* corresponde a observaciones sobre el estado del sistema a diagnosticar. En su forma más simple son observaciones sobre el estado actual, por ejemplo `temperatura(paciente) = 37`. También pueden recoger la evolución en el tiempo reciente de ciertos parámetros descriptivos en forma de lista, por ejemplo:

```
{temperatura(paciente, día-1) = 40,
 temperatura(paciente, día-2) = 37,
 temperatura(paciente, día-3) = 39}
```

- *síntomas:* un síntoma es una observación sobre el estado del sistema que indica cierta anormalidad o situación no deseada, por ejemplo: `temperatura (paciente) = 40`, `olor(motor) = quemado`, etc.

- *causas*: son los hechos que justifican la presencia de los síntomas. Las causas pueden ser *intermedias*, si son causas que pueden ser explicadas por causas más profundas, o *finales* en caso contrario. Las causas se dividen también en *internas* si son causas originadas en el propio sistema (por ejemplo la batería agotada en un vehículo o una enfermedad de un paciente) o *externas* si corresponden a influencias externas al sistema (por ejemplo, mala alimentación en un paciente o mantenimiento defectuoso de un equipo mecánico). Las causas internas se expresan habitualmente de la siguiente forma:

  a) estados del sistema que se diagnostica, por ejemplo, `enfermedad (paciente)= malaria`,

b) componentes averiados que presentan un fallo de comportamiento, por ejemplo, `componente-averiado(vehículo) = batería`,

c) componentes averiados a los que, además, se les asocia el estado concreto de avería que presentan, por ejemplo, `componente-averiado(vehículo) = batería(descargada)`

El problema de diagnóstico se define de la siguiente forma:

> **Definición 4.1:** *Diagnosticar* un sistema dinámico consiste en encontrar las causas que justifican la presencia de síntomas.

Habitualmente el proceso de diagnóstico encuentra las causas internas que explican los síntomas observados y, en algunas ocasiones, además se generan las causas externas. De forma resumida, los pasos principales para llevar a cabo diagnóstico son los siguientes:

1. *Detectar síntomas*. Dadas unas observaciones sobre un sistema, en este primer paso se seleccionan las que corresponden a situaciones no deseadas. Dicha detección puede plantearse de diversas formas:

   a) En un contexto de diagnóstico en donde se trata de identificar los componentes averiados, la detección de síntomas puede basarse en hacer una predicción de los valores que deberían observarse asumiendo comportamiento correcto y, mediante comparación, se señalan los que corresponden a síntomas.

   b) En un contexto de gestión de un sistema dinámico, los objetivos de gestión indican los criterios en los que se basa la detección de síntomas. Por ejemplo, en la gestión en tiempo real de una red de autovías en una zona

urbana uno de los objetivos principales (además de la seguridad vial) es mantener la fluidez del tráfico evitando que la velocidad media sea inferior a ciertos valores (por ejemplo 50 Km/h). La presencia de valores inferiores a dichos umbrales se entienden como síntomas que deben ser detectados y, después, analizados con el fin de encontrar soluciones. En general, los objetivos se pueden expresar en forma de valores deseados para ciertas observaciones (intervalos numéricos o valores cualitativos).

2. *Generar hipótesis*. En un segundo paso se generan hipótesis de causas que explican los síntomas.

3. *Discriminar hipótesis*. Si existen varias causas alternativas que explican los mismos síntomas es necesario solicitar información adicional sobre nuevas observaciones del sistema que permitan discriminar entre hipótesis. Tras obtener dicha información, el proceso se repite de forma iterativa hasta alcanzar un conjunto de hipótesis satisfactorio.

4.2 Organización del conocimiento

El diagnóstico basado en modelos lleva a cabo la resolución del problema mediante el manejo de un modelo del funcionamiento del sistema a diagnosticar. Dicho modelo, al menos, se puede representar mediante dos opciones alternativas:

a) relaciones causa-efecto (modelo causal),

b) conjunto de componentes interrelacionados (modelo de componentes)

En este apartado se describen ambas opciones mostrando la forma de organización del conocimiento que corresponde a cada una.

4.2.1. Modelo representado con relaciones causa-efecto

El modelo que se describe en este apartado corresponde a un método denominado *cubrir y diferenciar* que puede considerarse un caso particular del enfoque denominado *diagnóstico abductivo*, es decir, diagnóstico que se realiza razonando con un conjunto de relaciones causa-efecto desde los efectos hacia las causas. El método asume que los síntomas son datos de entrada, es decir no hay fase de detección de síntomas, y simula el proceso razonamiento de diagnóstico en dos pasos principales que se realizan de forma iterativa: un primer paso para generación de hipótesis de causas que explican síntomas observados y un segundo paso para discriminación de hipótesis de causas mediante la petición de observaciones adicionales.

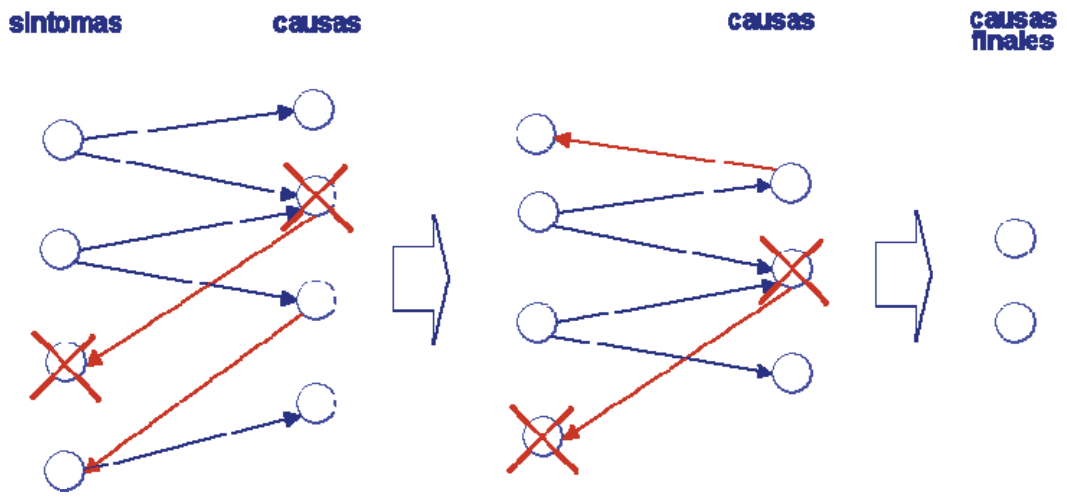


Figura 4.1: Visión general del proceso realizado por el método de cubrir y diferenciar.

Tipo/subtipo de conocimiento		**Explicación**	
Relaciones efectos-causas		*Significado*	Indican para cada síntoma (o causa intermedia) las posibles causas que lo explican
		Representación	Reglas
		Ejemplos	`nivel(transaminasa-GOT) = alto → enfermedad(paciente) = hepatitis`
Diferenciación	Relaciones causas-efectos	*Significado*	Indican la necesidad de observar un cierto hecho para cuando se presente una cierta causa
		Representación	Reglas
		Ejemplos	`depósito-gasolina(vehículo) = vacío → marcador-gasolina(vehículo) = cero`
	Conocimiento circunstancial	*Significado*	Credibilidad y preferencia de las hipótesis independientemente de los síntomas teniendo en cuenta únicamente observaciones circunstanciales
		Representación	Reglas, tablas de probabilidad
		Ejemplos	`localización(paciente)=tropical →PREFERIR(enfermedad(paciente)=malaria, enfermedad(paciente)=gripe)`
	Refino	*Significado*	Indica la necesidad de refinar un síntoma preguntando información de más detalle sobre dicho síntoma, lo que permite discriminar mejor entre un grupo de hipótesis.
		Representación	Reglas
		Ejemplos	`estado(motor) = no-arranca y avería(vehículo) = {batería-descargada, fallo-carburador, depósito-vacío} → PREGUNTAR(sonido(motor-arranque))`
	Cualificación	*Significado*	Condiciones que pueden anular relaciones efectos-causas.
		Representación	Reglas
		Ejemplos	`fiebre(paciente)=alta y nivel(PSA)=alto → NO-EVOCA(fiebre(paciente)=alta, enfermedad=infección-vírica)`
Estrategias de combinación		*Significado*	Excepciones al principio de parquedad para generación de combinaciones
		Representación	Reglas
		Ejemplos	`exceso(aire)=bajo y ajuste(radiación) = desequilibrio y ajuste(convención) = desequilibrio → COMBINAR (ajuste(radiación)=desequilibrio, ajuste(convención)=desequilibrio)`

Figura 4.2. Organización del conocimiento.

El sistema de diagnóstico médico especializado en medicina interna INTERNIST [Pople, 81] puede considerarse una versión inicial del método cubrir y diferenciar. Dicho sistema comenzó siendo DIALOG [Pople et al., 75], pasó por versiones de INTERNIST [Pople, 77; Miller et al., 82] y evolucionó después como CADUCEUS [Pople, 1982]. No obstante, la versión que se describe en este capítulo corresponde al caso descrito por la herramienta de adquisición del conocimiento MOLE [Eshelman, 88] que fue aplicado a la construcción de un sistema de diagnóstico de averías en una central térmica. El sistema CHECK [Console, Torasso, 90] es otro ejemplo de diagnóstico con modelo causal.

El método se basa en representar un modelo causal del funcionamiento del sistema a diagnosticar, sin necesidad de describir su estructura. En concreto, se trata de representar: (1) conocimiento sobre qué síntomas hacen sospechar ciertas causas y (2) conocimiento que permite descartar y/o confirmar hipótesis de causas. El razonamiento consiste en realizar un proceso iterativo realizando una inferencia de los síntomas hacia las causas para generar hipótesis y, después, una inferencia de las causas hacia los síntomas para discriminar hipótesis.

La figura 4.1 muestra una visión general del proceso de búsqueda realizado por el método. De acuerdo con dicha figura, a partir de un conjunto de síntomas (nodos de la izquierda de la figura) el método busca causas que los explican. Para discriminar entre dichas causas se pregunta la presencia de otros síntomas cuya ausencia (tachado en la figura) puede hacer que implique la eliminación de ciertas hipótesis de causas. Los síntomas por los que se preguntan pueden hacer sospechar nuevas causas, de forma que este proceso hacia las causas y hacia los síntomas se repite hasta que no quedan más síntomas por preguntar. A continuación, se genera una combinación verosímil de causas y el proceso se repite buscando por causas más profundas. Estos pasos se dan hasta que se llega al conjunto de causas finales que explican los síntomas observados. El proceso general, en vez de un proceso lineal que avanza a lo largo de distintos niveles de profundidad de causas, debe

contemplarse mejor como una búsqueda en árbol, dado que puede haber varias opciones alternativas de combinaciones de causas sobre las que profundizar.

El conocimiento del método cubrir y diferenciar es de tres tipos: conocimiento de relaciones efectos-causas, conocimiento de diferenciación y estrategias de combinación. El conocimiento de *relaciones efectos-causas* incluye relaciones que indican para cada síntoma (o causa intermedia) las posibles causas que lo explican. Un ejemplo de relación efecto-causa para un caso de medicina es el siguiente:

```
nivel(transaminasa-GOT-en-sangre) = alta <síntoma>
→ enfermedad(paciente) = hepatitis <hipótesis de causa>
```

La relación indica la posibilidad en la existencia de la causa, es decir no implica su presencia forzosa, sino que se plantea como hipótesis. Para denominar este tipo de relaciones utiliza en ocasiones el término *evoca*(*síntoma Si, causa Cj)*. En general, para un síntoma habrá varias hipótesis de causas (entendidas en forma disyuntiva) de las cuales se asume que al menos una de ellas será la que deba estar presente. Se asume *exhaustividad* para este tipo de conocimiento, es decir, todo síntoma debe tener asociado al menos una hipótesis de causa.

El conocimiento de *diferenciación* permite separar y elegir las hipótesis que explican los mismos síntomas. Para ello se utilizan los siguientes sub-tipos de conocimiento: relaciones causas-efectos, conocimiento circunstancial, conocimiento de refino y conocimiento de cualificación.

El *conocimiento de relaciones causas-efectos* consta de relaciones en donde se indica la obligación de observar un cierto efecto para cuando se presente una cierta causa. Complementa al conocimiento de relaciones efectos-causas pero se trata de relaciones diferentes dado que el hecho de que un efecto haga sospechar una causa no es lo mismo que, dada una causa, tenga que presentarse necesariamente un efecto. Para denominar este tipo de relaciones se suele utilizar el término

manifiesta(causa Ci, síntoma Sj). Debido a que estas relaciones expresan los efectos que necesariamente deben ser observados, la ausencia de dichos efectos permite descartar las correspondientes hipótesis de causas. Por tanto se puede utilizar como guía para determinar los datos adicionales a interrogar al usuario dirigidas a confirmar o descartar una hipótesis.

Por ejemplo, si un vehículo no arranca se puede considerar como hipótesis que no tiene gasolina. Si fuera cierto entonces necesariamente debería observarse que la aguja del marcador de la gasolina está a cero. Al preguntar por el marcador, si la respuesta es que no está a cero, entonces se puede descartar dicha hipótesis para considerar otras. La forma de representar este tipo relaciones puede ser también con reglas, por ejemplo::

```
depósito-gasolina(vehículo) = vacío <causa>
→ marcador-gasolina(vehículo) = cero <observación>
```

El *conocimiento circunstancial* (también llamado de refuerzo/debilitamiento de hipótesis o de cualificación de hipótesis) está asociado a la credibilidad a priori de las hipótesis independientemente de los síntomas que cubran. Es un conocimiento de naturaleza heurística que indica preferencia o descarte de hipótesis basada en información de contexto. La utilidad de esta información está en que, entre varias hipótesis posibles que explican los mismos síntomas, normalmente se prefieren las que son más probables ante ciertas circunstancias. Por ejemplo, en el dominio de medicina se puede considerar que si el paciente se encuentra en una zona tropical puede ser más probable que, ante los mismos síntomas, son mucho más probables un tipo de enfermedades (malaria, por ejemplo) frente a otras (gripe, por ejemplo). Se puede representar mediante reglas en cuyo antecedente se indica una observación que condiciona la preferencia entre varias hipótesis o el descarte de alguna hipótesis, por ejemplo:

```
sexo(paciente) = femenino <observación circunstancial>
→ NO(enfermedad(paciente)=tumor-en-próstata) <diferenciación>

localización(paciente)=tropical <observación circunstancial>
→ PREFERIR(enfermedad(paciente)=malaria,
           enfermedad(paciente)=gripe) <diferenciación>
```

El *conocimiento de refino* tiene como propósito indicar la necesidad de refinar un síntoma con información de más detalle para poder discriminar entre hipótesis. Por ejemplo, el síntoma coche-no-arranca (que lo explican tres hipótesis batería-descargada, fallo de carburación o depósito vacío) puede refinarse con detalles sobre si suena el motor durante el intento de arranque lo que permitiría discriminar la hipótesis de batería-descargada. Se trata de relaciones en donde dado un grupo de hipótesis se indica qué detalles sobre un síntoma es necesario conocer.

El *conocimiento de cualificación* (también llamado de refuerzo/debilitamiento de relaciones o de cualificación de relaciones) tiene que ver con la confianza de la relación evocar en cuanto a que puede estar condicionada con conocimiento adicional. Esto es útil para evitar que haya conjuntos de síntomas que se explican independientemente. Por ejemplo el síntoma S1 puede evocar la causa C1 pero al darse S2 es preferible descartar la causa C1 para centrarse en otras. En el campo de medicina, por ejemplo, un síntoma puede hacer sospechar una enfermedad leve pero la presencia de otro síntoma sobre una enfermedad grave hace que se elimine dicha sospecha. Estos criterios pueden contemplarse como una forma de meta-conocimiento dado que la observación de ciertos síntomas hace inhibir ciertas relaciones. Puede representarse en forma de reglas en donde en el antecedente se indican observaciones o síntomas y en el consecuente una relación efecto causa que debe ser anulada.

Respecto al conocimiento de *estrategias de combinación*, el método de cubrir y diferenciar asume dos criterios: exhaustividad y parquedad (*parsimony* en inglés). El primer criterio establece que se consideran únicamente combinaciones que explican *todos* los síntomas observados. El criterio de parquedad establece que se eligen conjuntos mínimos que explican los síntomas observados. Así, según la figura 4.3, dados los síntomas {S1, S2, S3, S4, S5} los conjuntos siguientes cumplen el principio de exhaustividad, es decir, cubren todos los síntomas observados {C1, C3}, {C1, C2, C3}, {C1, C2, C4}, {C1, C3, C4} y {C1, C2, C3, C4}. Sin embargo, aplicando el principio de parquedad sólo se consideran como válidos los conjuntos {C1, C3} y {C1, C2, C4} dado que los otros son superconjuntos de éstos.

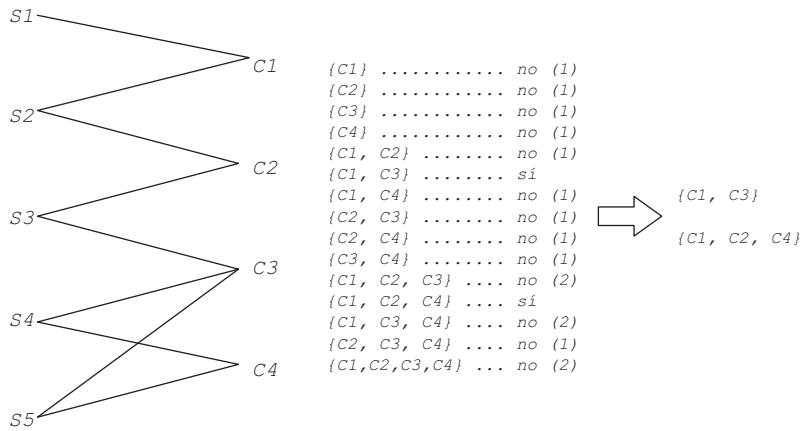


Figura 4.3: Ejemplo del proceso de generación de combinaciones. En la figura se marcan con (1) las combinaciones que no satisfacen el criterio de exhaustividad y con (2) las que no satisfacen el criterio de parquedad.

Teniendo en cuenta estos principios, el conocimiento de estrategias de combinación se utiliza para expresar *excepciones* a la regla anterior como es el caso de que un determinado síntoma deba ser explicado por dos causas a la vez. Por ejemplo baja transferencia de calor puede ser explicada por desequilibrio en radiación y desequilibrio en convención. Sin embargo, si se da bajo exceso de aire, se deben contemplar simultáneamente ambas causas. Por lo que la regla de combinación dice: cuando se da bajo exceso de aire y se tienen como causas desequilibrio en radiación y convención, debe generarse una combinación con ambas causas.

4.2.2. Modelo representado mediante componentes

El enfoque sobre diagnóstico que se presenta en este apartado pertenece al planteamiento general denominado diagnóstico basado en consistencia (*consistency-based diagnosis)* que fue inicialmente propuesto para utilizar modelos que contienen axiomas sobre el comportamiento correcto del sistema. El diagnóstico en este caso se basa en la realización de un contraste (análisis de consistencia) entre las medidas derivadas de dichos axiomas y las medidas observadas en el sistema a diagnosticar. Las discrepancias entre ambas son la base de la búsqueda de los posibles fallos. El concepto de diagnóstico basado en la consistencia podría considerarse también para tratar también modelos de fallos, es decir, axiomas correspondientes a estados de avería de componentes.

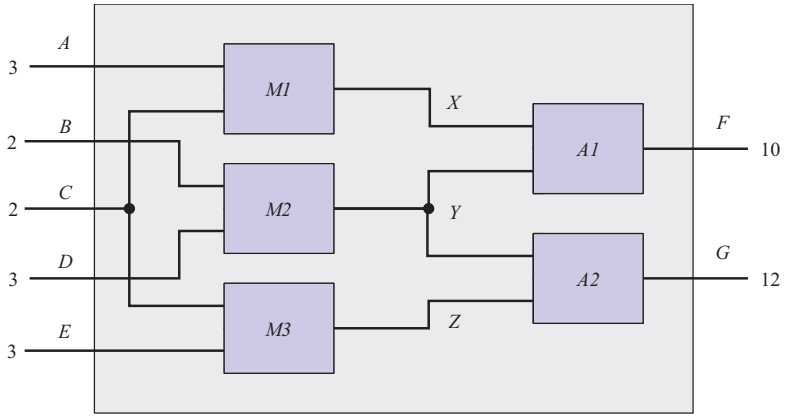


Figura 4.4: Ejemplo de circuito digital para diagnóstico [de Kleer, Williams, 87].

El enfoque que se describe aquí es aplicable a diagnóstico de dispositivos o sistemas físicos de los que se conoce con detalle la estructura y funcionamiento de sus componentes. Por ejemplo, sistemas como circuitos digitales, sistemas eléctricos, plantas químicas, etc. Para ilustrar la descripción de la organización del conocimiento se va utilizar principalmente el caso del sistema GDE (*General Diagnostic Engine*) [de Kleer, Williams, 87]. En la figura 4.4 se muestra un circuito digital sencillo con tres multiplicadores, M1, M2 y M3, y dos sumadores A1, A2, conectados tal como se observa en dicha figura.

Tipo/subtipo de conocim.		**Explicación**	
Estructura	Componentes	*Significado*	Tipos de componentes y lista de componentes que forman la estructura interna del sistema.
		Representación	Conjuntos, predicados lógicos
		Ejemplos	`sumador(M1), sumador(A2), multiplicador(M1)`
	Interconexión	*Significado*	Descripción la interconexión entre componentes.
		Representación	Predicados lógicos, grafos
		Ejemplos	`conexión(out(1, M1),in(1, A2))`
	Niveles de abstracción	*Significado*	Organización jerárquica de componentes con relación *es-parte-de*.
		Representación	Predicados lógicos, grafos
		Ejemplos	`componentes(circuito-general,{circuito-a, circuito-B}).` `componentes(circuito-A,{A2, A3})`
Comporta-miento	Modelo directo	*Significado*	Descripción del comportamiento de cada componente bajo la suposición de que funciona correctamente
		Representación	Fórmulas lógicas, restricciones, reglas
		Ejemplos	`sumador(A), val(in(1,A), X), val(in(2,A),Y)` → `val(out(1,A),X+Y)`
	Modelo inverso	*Significado*	Descripción del comportamiento de los componentes en sentido opuesto al que opera realmente
		Representación	Fórmulas lógicas, restricciones, reglas
		Ejemplos	`sumador(a), val(in(1,a), x), val(out(1,a),y)` → `val(in(2,a), y-x)`
	Modelo de fallos	*Significado*	Descripción del funcionamiento de los componentes ante la presencia de diversos fallos
		Representación	Fórmulas lógicas, restricciones, reglas
		Ejemplos	`temperatura(1, creciente), temperatura(2, creciente), estado(componente-A, avería-3)` → `presión(3, decreciente)`
Probabilidad de fallos		*Significado*	Probabilidad de fallo de cada uno de los componentes.
		Representación	Medidas de probabilidad asociadas a fallos de componentes de forma individual.
		Ejemplos	`P(componente-5, avería-3) = 0.000005` `P(componente-9, avería-1) = 0.00000003`
Preferencias de adquisición		*Significado*	Preferencias sobre formas de adquisición de nuevas observaciones
		Representación	Medidas numéricas de coste de adquisición asociadas a observaciones, o bien reglas indicando preferencias

		Ejemplos	`tipo(X)=análisis y dolor(X)=presente`
			`tipo(Y)=análisis y dolor(Y)=ausente`
			→ `PREFERIR(Y,X)`

Figura 4.5. Organización del conocimiento.

La figura 4.5 resume los diversos tipos de conocimiento que se consideran para este método. En primer lugar se tiene lo que se denomina *estructura* que consiste en una descripción de la estructura interna del sistema. Esto normalmente incluye la definición de los tipos de componentes y la enumeración de cada uno de ellos. Esto podría representarse, por ejemplo, mediante predicados lógicos con constantes de la siguiente forma:

```
multiplicador(M1)
multiplicador(M2)
multiplicador(M3)
sumador(A1)
sumador(A2)
```

Por otro lado debe indicarse la *interconexión* entre componentes. Esta información se puede representar con ayuda de un predicado de la siguiente forma (en donde *out(1, M1)* indica la salida número 1 del componente M1, *in(1, A1)* se refiere a la entrada número 1 del componente A1):

```
conexión(out(1,M1), in(1,A1))
conexión(out(1,M2), in(2,A1))
conexión(out(1,M2), in(1,A2))
conexión(out(1,M3), in(2,A2))
```

Además, opcionalmente, la estructura del sistema se puede formular a diferentes *niveles de abstracción* de forma que el sistema se puede contemplar formado por un conjunto de componentes de alto nivel que a su vez están formados por subcomponentes. Así, la estructura se puede organizar en una jerarquía de componentes en donde la raíz el sistema completo y las hojas son los componentes más sencillos. Esto puede representarse de una forma similar a los anteriores ejemplos:

```
componentes(circuito-general,{circuito-A, circuito-B}).
componentes(circuito-A,{M1, M2, M3})
componentes(circuito-B,{A1, A2})
```

Por otra parte, el modelo de *comportamiento* es una descripción de cómo se comportan cada uno de los componentes que forma el sistema. La descripción de la estructura y comportamiento debe satisfacer el principio denominado *no función en la estructura*, que indica que el comportamiento de los componentes debe describirse localmente, es decir, independientemente del contexto (función) en donde opera [de Kleer, Brown, 84]. Por ejemplo, la descripción del comportamiento de un interruptor no se debe expresar indicando que permite apagar o encender una bombilla (esto es la función) sino interrumpir o permitir el paso de corriente eléctrica entre dos puntos. El comportamiento de los componentes se puede expresar con las siguientes expresiones lógicas (en donde `A`, `X` e `Y` son variables y `val(in(1,A),X)` expresa que `X` es la entrada número 1 del componente `A`):

```
sumador(A), val(in(1,A),X), val(in(2,A),Y) ->
val(out(1,A),X+Y))

multiplicador(A), val(in(1,A),X), val(in(2,A),Y) ->
val(out(1,A),X*Y))
```

Este tipo de representación puede procesarse mediante motor de inferencia que utilice un mecanismo de deducción automática para generar respuestas sobre los valores que deben observarse en diversos puntos del circuito. El modelo de comportamiento puede expresar la forma de funcionamiento de los componentes en sentido directo, es decir, desde las entradas hacia las salidas. Pero en ocasiones es posible indicar también el funcionamiento en sentido opuesto. Esto se recoge en lo que se denomina *modelo inverso*. Por ejemplo, el modelo inverso de un sumador puede expresarse con las siguientes dos sentencias:

```
sumador(A), val(in(1,A),X), val(out(1,A),Y) ->
val(in(2,A),Y - X))
sumador(A), val(in(2,A),X), val(out(1,A),Y) ->
val(in(1,A),Y - X))
```

Las dos sentencias anteriores junto a la que expresa el comportamiento en sentido directo describe el comportamiento completo del sumador en ambos sentidos. Como alternativa a esta representación es posible también utilizar reglas o restricciones. Las restricciones correspondientes al circuito de la figura 4.4 serían:

```
M1: X = A * C
M2: Y = B * D
M3: Z = C * E
A1: F = X + Y
A2: G = Y + Z
```

Las restricciones pueden procesarse mediante procedimientos de satisfacción de restricciones para obtener los valores de unas variables a partir de los valores de otras, lo que permite hacer una predicción de los efectos que se esperan a partir de unas causas conocidas (sentido directo) o una estimación de las causas necesarias para unos determinados efectos (sentido inverso). Las restricciones representan de una forma más concisa el comportamiento incluyendo en una única sentencia información del comportamiento directo e inverso, aunque exigen mayor coste de proceso.

El modelo de comportamiento puede incluir conocimiento sobre la presencia de fallos típicos. Esto se recoge en lo que se denomina *modelo de fallos* que indica el comportamiento de cada componente bajo el supuesto de que presentan ciertas averías típicas. Para cada componente y estado de avería se expresa el conocimiento

que permite determinar un conjunto de valores en efectos a partir de unas determinadas causas como muestra el siguiente caso:

```
temperatura(A1, creciente), temperatura(A2, creciente),
estado(componente-A, avería-3) → presión(A3, decreciente)
```

Opcionalmente, también se puede tener un modelo inverso del modelo de fallos que permita razonar desde los efectos a las causas. Para cada componente se puede indicar la probabilidad de fallo. Este es un dato con el que en ocasiones suministra el fabricante del componente en forma de medida de probabilidad. Además, es posible contar también con datos sobre el coste de adquisición de nuevas observaciones, que se puede expresar de forma numérica o bien con una ordenación (total o parcial). Aquí, se puede también hacer uso de una representación con reglas para indicar condiciones en las preferencias sobre la adquisición de unas observaciones sobre otras.

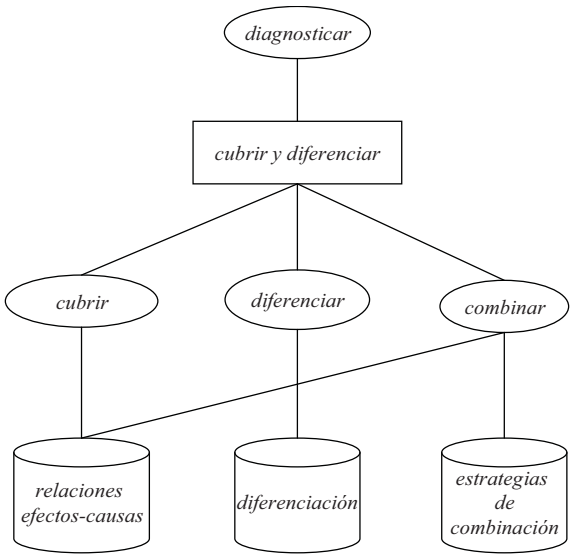


Figura 4.6: Estructura del método considerada en el algoritmo de cubrir y diferenciar.

Rol dinámico	Significado	Representación	Ejemplo
observaciones	datos observados utilizados como información de entrada del problema	conjunto de ternas concepto-atributo-valor	{transaminasa-GPT(paciente)=alta, transaminasa-GOT(paciente)=alta }
eventos	eventos a explicar (tanto datos observados como causas intermedias)	conjunto de ternas concepto-atributo-valor	{transaminasa-GPT(paciente)=alta, transaminasa-GOT(paciente)=alta }
hipótesis-de-causas	conjunto de hipótesis de causas	conjunto de ternas concepto-atributo-valor	{enfermedad(paciente)=hepatitis, enfermedad(paciente)=tumor, alimentación(paciente)=desequilibrada}
combinaciones	combinaciones de conjuntos de hipótesis de causas	conjunto de conjuntos de ternas concepto-atributo-valor (disyunción de conjunciones)	{{enfermedad(paciente)=tumor, paciente-alimentación(paciente)=desequilibrada}, {enfermedad(paciente)=hepatitis, alimentación(paciente)=desequilibrada}}
causas-finales	soluciones posibles del proceso de diagnóstico	conjunto de conjuntos de ternas concepto-atributo-valor (disyunción de conjunciones)	{{enfermedad(paciente)=tumor, alimentación(paciente)=desequilibrada}, {enfermedad(paciente)=hepatitis, alimentación(paciente)=desequilibrada}}

Figura 4.7: Roles dinámicos que intervienen en el proceso de inferencia de cubrir y diferenciar.

4.3 Inferencia

4.3.1. Algoritmo 1: Cubrir y diferenciar

Los pasos de inferencia del método de cubrir y diferenciar son: cubrir, diferenciar y combinar. El paso de inferencia *cubrir* tiene como objetivo generar hipótesis de causas a partir de eventos (tanto observaciones como causas intermedias) utilizando el conocimiento de relaciones efectos-causas. Para ello, para cada evento se busca en la base de conocimiento qué hipótesis causa genera. Si un evento es causa final se genera también como salida indicando que se explica a sí mismo (esto simplifica el algoritmo).

```
INFERENCIA cubrir
       DATOS:                 eventos
       BASES DE CONOCIMIENTO: relaciones-efectos-causas
       RESULTADOS:            hipótesis-de-causas

INFERENCIA diferenciar
       DATOS:                 hipótesis-de-causas, observaciones
       BASES DE CONOCIMIENTO: conocimiento-de-diferenciación
       RESULTADOS:            hipótesis-de-causas, eventos

INFERENCIA combinar
       DATOS:                 hipótesis-de-causas, observaciones
       BASES DE CONOCIMIENTO:relaciones-efectos-causa, estrategias-combinación
       RESULTADOS:            combinaciones
```

Figura 4.8: Pasos de inferencia considerados en el algoritmo de cubrir y diferenciar.

El objetivo del paso de inferencia *diferenciar* es elegir unas hipótesis de causa frente otras, de acuerdo con el conocimiento de diferenciación, solicitando eventualmente nueva información. Para ello, se aplican los criterios de cada una de las bases de conocimiento. En particular, la base de conocimiento de causas-efectos se utiliza para preguntar nuevos datos que, en caso de no presentarse, rechazan la hipótesis considerada. Por ejemplo, tal como se ilustró previamante, supóngase que se tiene la hipótesis de causa *depósito-gasolina(vehículo) = vacío* y la regla de la base de causas-efectos:

```
depósito-gasolina(vehículo) = vacío
→ marcador-gasolina(vehículo) = cero
```

Aplicando dicha regla, el paso de inferencia interroga al usuario para comprobar que se tiene la observación *marcador-gasolina(vehículo) = cero*. Si la respuesta del usuario es que el marcador de gasolina es diferente de cero, entonces se elimina del conjunto de hipótesis de causas la hipótesis *depósito-gasolina(vehículo) = vacío*. Si el usuario no conoce la respuesta la hipótesis de causa se mantiene dado que no se dispone de información para rechazarla.

La base de conocimiento circunstancial permite también rechazar hipótesis utilizando criterios de preferencia bien de forma individual o bien mediante análisis conjunto de la presencia de varias hipótesis. Por ejemplo, considerar que se tiene la hipótesis de causa `enfermedad(paciente)=tumor-en-próstata` y la regla en la base de conocimiento circunstancial:

```
sexo(paciente) = femenino
→ NO(enfermedad(paciente)=tumor-en-próstata)
```

El paso de inferencia interroga al usuario para conocer el sexo del paciente (observación circunstancial) de forma que, en caso de que sea femenino, la hipótesis se elimina del conjunto de causas posibles.

Por ejemplo, considerar que se tienen las causas `enfermedad(paciente)=malaria` y `enfermedad(paciente)=gripe` y la regla en la base de conocimiento circunstancial:

```
localización(paciente)=tropical
→ PREFERIR(enfermedad(paciente)=malaria,
           enfermedad(paciente)=gripe)
```

En este caso, el paso de inferencia interroga al usuario para conocer la localización del paciente (observación circunstancial) y, en caso de que ser tropical se elimina la causa de gripe del conjunto de causas.

En ocasiones, según lo determine las características del dominio del problema, el paso de inferencia debe realizar una validación antes de eliminar causas aplicando conocimiento circunstancial. La validación consiste en que para preferir una causa A frente a otra causa B, B sólo podrá ser eliminada si para cada síntoma observado que explica B existe al menos otra hipótesis de causa.

El conocimiento de refino se utiliza también para interrogar al usuario para solicitar información adicional que permita rechazar alguna hipótesis. La base de cualificación se utiliza para eliminar relaciones de cubrimiento (por simplificación

para facilitar la presentación del método, este caso no se ha contemplado aquí dado que, en principio, exige el mantenimiento de la red explicativa como estructura de datos que se va modificando conforme actúan las reglas).

Los datos que recibe el paso de inferencia de diferenciación son el conjunto de hipótesis a diferenciar junto con las observaciones conocidas hasta el momento. Como resultado se genera un nuevo conjunto de hipótesis que contiene las hipótesis que se mantienen y las nuevas observaciones que se han adquirido.

```
METODO cubrir-y-diferenciar
  DATOS: observaciones
  RESULTADOS: causas-finales

ALGORITMO
1.   combinaciones := ϕ
2.   explicar(observaciones -> causas-finales)

PROCEDIMIENTO explicar
  DATOS: eventos
  RESULTADOS: causas-finales
1.   hipótesis-de-causas := ϕ
2.   REPEAT
3.     cubrir(eventos -> H)
4.     hipótesis-de-causas := hipótesis-de-causas U H
5.     diferenciar(hipótesis-de-causas, observaciones ->
6.                 hipótesis-de-causas, eventos)
7.     observaciones := observaciones U eventos
8.   UNTIL eventos = ϕ
9.   propagar(hipótesis-de-causas -> causas-finales)

PROCEDIMIENTO propagar
  DATOS: hipótesis-de-causas
  RESULTADOS: causas-finales
1.    combinar(hipótesis-de-causas, observaciones -> combinaciones)
2.    REPEAT
3.      GET(eventos, combinaciones)
4.      IF eventos sólo contiene causas finales
5.        THEN causas-finales := causas-finales U {eventos}
6.        ELSE explicar(eventos -> causas-finales)
7.    UNTIL combinaciones = ϕ
```

Figura 4.9: Ejemplo de algoritmo del método cubrir y diferenciar

Finalmente, el paso de inferencia de *combinar* aplica el criterio de parquedad para obtener combinaciones de causas que explican todos los síntomas, aplicando las excepciones que forman parte de la base de conocimiento de estrategias de combinación.

Esta versión de algoritmo del método cubrir y diferenciar asume el supuesto de que, en los caminos desde los síntomas a las causas, no se produce repetición de causas de diferentes niveles tal como ilustra el siguiente ejemplo. Supóngase que tras combinar un conjunto de causas se obtiene {A,B}. Después, dichas causas a su vez evocan {A, C, D} lo que significa que vuelve a aparecer A por ejemplo porque desde B se evoca a A. En caso de que sea necesario tratar esta situación en un determinado dominio, el algoritmo debería ser adaptado.

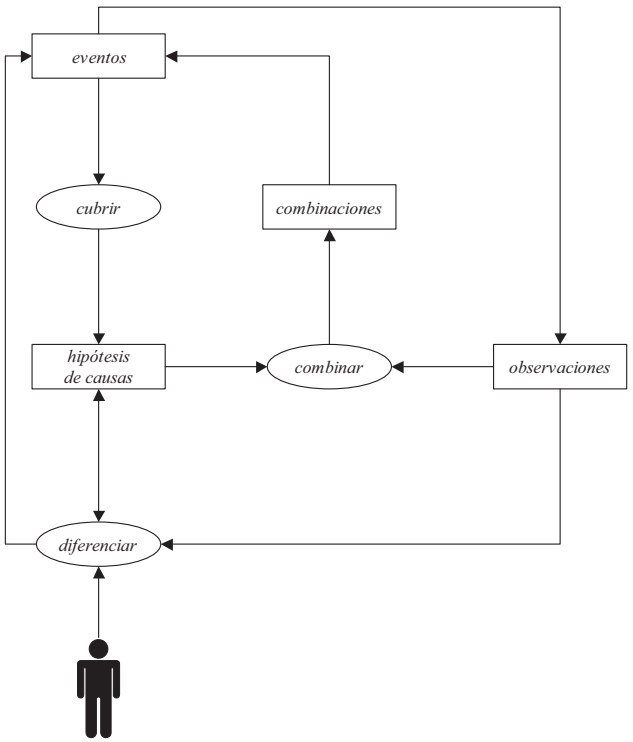


Figura 4.10: Estructura de inferencia del ejemplo de algoritmo.

4.3.2. Algoritmo 2: Diagnóstico basado en modelo de componentes

Para mostrar el algoritmo de diagnóstico basado en modelo de componentes se hará uso del ejemplo de la figura 4.4. El algoritmo descrito aquí corresponde al caso del sistema GDE siguiendo la simplificación descrita en [Forbus y De Kleer, 93]. El modelo de comportamiento se asume que está representado mediante restricciones:

```
M1: X = A * C
M2: Y = B * D
M3: Z = C * E
A1: F = X + Y
A2: G = Y + Z
```

Este modelo permite predecir las medidas que se esperan tener bajo el supuesto de un funcionamiento correcto junto a las explicaciones de los componentes que las producen. Por ejemplo, en el ejemplo descrito, si se sabe que A = 3, B = 2, C = 2, D = 3 y E = 3 un proceso de satisfacción de restricciones generaría los valores para el resto de variables junto a las explicaciones correspondientes:

```
X = 6  (M1: A=3, C=2)
Y = 6  (M2: B=2, D=3)
Z = 6  (M3: C=2, E=3)
F = 12 (A1: X=6, Y=6)
G = 12 (A2: Y=6, Z=6)
```

El análisis de las explicaciones permite determinar el conjunto de componentes en los que se basa la obtención de una determinada medida. Por ejemplo, la medida F=12 se basa en la operación correcta de los componentes {M1, M2, A1}. Esto permite construir lo que se denomina *estado* en un sistema de mantenimiento de la verdad ATMS [de Kleer, 86; Forbus, de Kleer, 93; Dressler, 88] indicando para cada medida su *entorno*, es decir las suposiciones sobre el correcto funcionamiento de los componentes que la generan.

Por ejemplo, el estado a partir de las medidas A, B, C, D, E es el siguiente (ver anexo en el presente libro sobre Sistemas de Mantenimiento de la Verdad para analizar con más detalle el presente ejemplo).:

```
JUSTIFICACIONES:
A=3, C=2, M1 → X=6
B=2, D=3, M2 → Y=6
C=2, E=3, M2 → Z=6
X=6, Y=6, A1 → F=12
Y=6, Z=6, A2 → G=12

ETIQUETAS:
<A=3,{{}}>
<B=2,{{}}>
<C=2,{{}}>
<D=3,{{}}>
<E=3,{{}}>
<X=6,{{M1}}>
<Y=6,{{M2}}>
<Z=6,{{M3}}>
<F=12,{{A1,M1,M2}}>
<G=12,{{A2,M2,M3}}>
```

En este ejemplo, el entorno de la medida A=3 es vacío dado que es una observación y no se apoya en el funcionamiento de componentes. La medida F=12 se explica por el funcionamiento correcto de los componentes A1, M1 y M2.

Si a continuación se observa la medida F=10 entonces se determina inmediatamente que el conjunto {A1, M1, M2} es un *conflicto*, es decir, un conjunto en donde al menos uno de los componentes falla. Esto hace que la etiqueta de la medida F=12 se elimine del estado y se inserte una sentencia denominada en la terminología de ATMS *no-good* correspondiente a entornos no consistentes. Además, a partir de F=10 es posible deducir otras medidas con las restricciones. Por ejemplo, con F=10 y X=6, haciendo uso de la restricción del componente A1 se obtiene Y=4. Esto da lugar a un nuevo estado:

```
JUSTIFICACIONES:
A=3, C=2, M1 → X=6
B=2, D=3, M2 → Y=6
C=2, E=3, M3 → Z=6
Y=6, Z=6, A2 → G=12
F=10,X=6, A1 → Y=4
F=10,Y=6, A1 → X=4
Y=4, Z=6, A2 → G=10

ETIQUETAS:
<A=3,{{}}>
<B=2,{{}}>
<C=2,{{}}>
<D=3,{{}}>
<E=3,{{}}>
<X=4,{{A1,M2}}>
<X=6,{{M1}}>
<Y=4,{{A1,M1}}>
<Y=6,{{M2}}>
<Z=6,{{M3}}>
<F=10,{{}}>
<G=10,{{A1,A2,M1,M3}}>
<G=12,{{A2,M2,M3}}>
<no-good,{{A1,M1,M2}}>
```

En este caso se obtienen dos predicciones para la medida G. Por un lado G=12 que se deriva de las medidas A, B, C, D y E procesando los cálculos hacia delante y, por otro lado, G=10 que se obtiene a partir de F=10 y otras medidas observadas mediante A1, A2, M1 y M3. Si a continuación se mide G=12 entonces se obtiene inmediatamente como conflicto {A1, A2, M1, M3}. Finalmente, el estado queda de la siguiente forma:

```
JUSTIFICACIONES:
A=3, C=2, M1 → X=6
B=2, D=3, M2 → Y=6
C=2, E=3, M3 → Z=6
F=10,X=6, A1 → Y=4
F=10,Y=6, A1 → X=4
G=12,Y=4, A2 → Z=8
G=12,Y=6, A2 → Z=6
G=12,Z=6, A2 → Y=6
```

```
ETIQUETAS:
<A=3,{{}}>
<B=2,{{}}>
<C=2,{{}}>
<D=3,{{}}>
<E=3,{{}}>
<X=4,{{A1,M2},{A1,A2,M3}}>
<X=6,{{M1}>
<Y=4,{{A1,M1}}>
<Y=6,{{M2},{A2,M3}}>
<Z=6,{{M3},{A2,M2}}>
<Z=8,{{A1,A2,M1}}>
<F=10,{{}}>
<G=12,{{}}>
<no-good,{{A1,M1,M2},{A1,A2,M1,M3}}>
```

El algoritmo considerado incluye tres pasos de inferencia: detectar, generar y adquirir. El paso de inferencia *detectar* tiene como fin determinar la presencia de síntomas y obtener los conflictos. Para ello utiliza el estado ATMS para encontrar los conflictos descritos previamente. En este ejemplo, se da como resultado los conflictos {A1, M1, M2} y {A1, A2, M1, M3}. Estos conjuntos son conflictos mínimos, es decir cualquier superconjunto de estos conjuntos es también conflicto.

```
INFERENCIA detectar
       DATOS:                 observaciones, estado
       BASES DE CONOCIMIENTO: estructura, comportamiento
       RESULTADOS:            conflictos, estado

INFERENCIA generar
       DATOS:                 conflictos
       BASES DE CONOCIMIENTO: -
       RESULTADOS:            candidatos

INFERENCIA discriminar
       DATOS:                 candidatos, estado
       BASES DE CONOCIMIENTO:-
       RESULTADOS:            observación
```

Figura 4.11: Pasos de inferencia considerados en el algoritmo.

El paso de inferencia *generar* tiene como fin proponer *candidatos* a partir de los conflictos. Un candidato es un conjunto de componentes que fallan y que explican la presencia de los síntomas observados. En el ejemplo, a partir de los dos conflictos {A1, M1, M2} y {A1, A2, M1, M3} se generarían los siguientes candidatos mínimos {A1}, {M1}, {A2, M2} y {M2, M3} es decir todos los conjuntos que cumplen dos condiciones: (1) presentan intersección no nula con todos los conflictos y (2) son mínimos, es decir, no se puede eliminar ningún elemento de dicho conjunto de forma que se siga cumpliendo la primera condición. Con el fin de reducir la complejidad computacional, el método que se describe aquí asume ciertas simplificaciones:

- los componentes fallan a priori con igual probabilidad,
- la probabilidad de fallo es un valor muy pequeño,
- el coste de realización de las observaciones es igual para todas las medidas,
- los componentes fallan de forma independiente.

Debido a estas hipótesis, es posible tener en cuenta únicamente candidatos de cardinalidad mínima, es decir, aquellos diagnósticos en donde falla el menor número posible de componentes. Así, si la cardinalidad mínima es *C*, entonces sólo es necesario considerar diagnósticos con un número de *C* componentes con fallo dado que cualquier diagnóstico con un número mayor de *C* tendrá una probabilidad significativamente menor. El método puede comenzar considerando diagnósticos de cardinalidad 1 aunque, en el proceso de eliminación de candidatos puede después considerar diagnósticos con cardinalidad superior. Según el ejemplo anterior los candidatos que se tienen son {A1}, {M1}, {A2, M2} y {M2, M3}. Los candidatos de cardinalidad mínima M=1 son {A1}, {M1}, por lo que el paso de inferencia *generar* da como resultado {{A1},{M1}}.

El paso de inferencia *discriminar* tiene como objetivo solicitar el valor de una nueva observación que permita discriminar entre los candidatos propuestos. Para ello, se busca la medida que pueda aportar mayor información y que tenga menor coste. En la versión del método que se plantea aquí con las simplificaciones que se han indicado, se utiliza una fórmula derivada del cálculo de la entropía según teoría de la información [Ben-Bassat 78]. Para una extensión y generalización de este método pueden consultarse las fuentes de [De Kleer, 90] y [De Kleer, Williams 87], en donde de forma general se plantea un enfoque probabilístico de selección de medidas basado en un modelo bayesiano. El procedimiento considerado supone que los candidatos con mínima cardinalidad tienen C componentes con fallo. La mejor medida M con valores V_i es aquella que minimiza un valor de preferencia que mide la entropía esperada $H_e(X)$ dado por la siguiente fórmula (ver ejemplo 3 en donde se explica la forma de deducirla):

$$H_e(M) = \sum N_i \ln N_i$$

En donde $N_i > 0$ es el número de candidatos de cardinalidad C que predicen el valor de la medida $M = V_i$. Normalmente, en este punto, muchos de los candidatos de tamaño mayor que C han sido eliminados por lo que hay un número pequeño de diagnósticos a considerar. La estrategia que se aplica aquí se denomina de un paso de anticipación (en inglés *one step look ahead*) que evalúa la forma en que se modifica la entropía para cada una de las medidas (de una en una) y se elige aquella medida que haga menor la entropía o, lo que es lo mismo, la medidas cuyos valores separen más los candidatos [1].

[1] El caso de que el número de candidatos sea $N_i = 0$ para algún valor de una medida $M = V_i$ significa que observar dicho valor no va a ayudar a distinguir entre los candidatos posibles. En este caso, una solución de tipo práctico podría ser considerar $N_i = T$, en donde T es el número total de candidatos de cardinalidad C.

En el ejemplo anterior se tenían como candidatos de mínima cardinalidad {A1} y {M1}. El estado ATMS correspondiente a las medidas X, Y y Z es:

```
<X=4,{A1,M2}{A1,A2,M3}>
<X=6,{M1}>
<Y=4,{A1,M1}>
<Y=6,{M2},{A2,M3}>
<Z=6,{M3},{A2,M2}>
<Z=8,{A1,A2,M1}>
```

Por ejemplo, para calcular $H_e(X)$ (la entropía para la medida X) se consideran cada una de sus valores posibles: X=4 y X=6. Para el caso del valor X=4, teniendo en cuenta que los dos candidatos son {A1} y {M1}, se consulta el estado ATMS para comprobar si predicen X=4. El candidato {A1} (que significa que A1 falla y el resto de componentes operan correctamente) no predice X=4 porque la etiqueta de X=4 en el ATMS es {{A1,M2}, {A1,A2,M3}}. Dicha etiqueta significa que para que se observe X=4 tienen que operar correctamente A1 y M2 o bien A1, A2 y M3, es decir, A1 tiene que funcionar bien necesariamente. El candidato {M1} (que significa que M1 falla y el resto de componentes operan correctamente) sí predice X=4. Como consecuencia, el conjunto S correspondiente a X=4 (que tiene los candidatos que predicen X=4) es S ={{M1}}. Como criterio general, un candidato C_j formará parte del conjunto S del valor de una medida $M = V_i$ si C_j tiene intersección vacía con al menos un conjunto de la etiqueta de $M = V_i$ en el estado del ATMS. Para el caso de X=6 el conjunto S es S={{A1}}. Aplicando la fórmula de cálculo de $H_e(X)$, teniendo en cuenta que número de elementos de S es 1 para X=4 y también es 1 para X=6, se tiene:

$$H_e(X) = 1 \ln 1 + 1 \ln 1 = 0$$

Siguiendo un procedimiento similar para calcular la entropía de las otras medidas Y y Z se tiene:

```
Y=6, S={{A1},{M1}}
Z=6, S={{A1},{M1}}
```

La aplicación de la fórmula de entropía para estas medidas es:

$$H_e(Y) = 2 \ln 2 = 1.4$$
$$H_e(Z) = 2 \ln 2 = 1.4$$

Como resultado de estos cálculos se obtiene que la medida X obtiene la entropía con valor más pequeño. Por tanto, X será elegida como siguiente medida a realizar.

Como se ha indicado, el algoritmo que se describe aquí realiza ciertos supuestos que limitan su aplicabilidad (probabilidad a priori igual de fallo de componentes con valor muy pequeño e independiente, además de que el coste de realización de las observaciones se asume igual para todas las medidas). Por ello, el algoritmo no aprovecha la posibilidad de que exista conocimiento de probabilidad de fallos ni de preferencias de adquisición que en algunos casos puede estar disponible. La generalización directa de este enfoque para ser aplicable a dichos casos presenta problemas de complejidad computacional dado que el proceso de diagnóstico basado en generación de candidatos puede dar lugar a una explosión combinatoria no manejable en problemas de cierta dimensión. Además puede plantear diagnósticos que aunque sean lógicamente posibles pueden ser físicamente poco relevantes.

Para resolver estas dificultades se han planteado variantes que extienden y/o mejoran el método aquí presentado tales como GDE+ [Struss, Dressler, 89] con modelos de fallo, Sherlock [De Kleer, Williams, 89] con modos de comportamiento

y un proceso de búsqueda que evita considerar candidatos no plausibles y XDE [Hamscher, 1991] con estructuras jerárquicas y aplicable a problemas más complejos. Para enriquecer el proceso de determinación de mediciones haciendo uso de enfoques que combinan entropía con el coste de la medición puede consultarse [Howard, 66]. Como alternativa al modelo basado en componentes se tiene el enfoque funcional [Sticklen et al. 89] que permite hacer más sencilla y eficiente la descripción del sistema, mediante abstracción de estructura y comportamiento sin seguir el principio *no función en estructura,* aunque con un planteamiento más dependiente de los dispositivos y, por tanto, menos general. Para una visión global de los diferentes métodos de diagnóstico pueden consultarse [Hamscher et al. 92] y [Benjamins, 93].

Rol dinámico	Significado	Representación	Ejemplo
`observacio-nes`	datos observados que se van actualizando conforme avanza el proceso de diagnóstico	conjunto de atributo-valor	`{A=3, B=2, C=2, D=3, E=3, F=10, G=12},`
`estado`	sentencias que forman la base de datos ATMS	conjunto de tuplas <atributo-valor, {conjunto-1, …, conjunto-N}>	`{<A=3,{}>, <B=2,{}>,<C=2,{}>, <D=3,{}>, <E=3,{}>, <X=6,{{M1}}>, <Y=6,{{M2}}>, <Z=8,{{M3}}>, <G=12,{{A2,M2,M3}}>, <F=12,{{A1,M2,M3}}>}`
`conflictos`	un conflicto es un conjunto de componentes entre los cuales falla al menos uno	conjunto de conjuntos de identificadores de componentes	`{{A1,M1,M2}, {A1,A2,M1,M3}}`
`candidatos`	un candidato es una hipótesis de conjunto de componentes que fallan	conjunto de conjuntos de identificadores de componentes	`{{A1}, {M1}, {A2, M2}, {M2, M3}}`
`componentes -averiados`	conjunto de componentes que fallan resultado del proceso de diagnóstico	conjunto de identificadores de componentes	`{A1,M2}`
`observación`	medida que es necesario conocer para continuar el proceso de diagnóstico	atributo-valor	`Z=6`

Figura 4.12: Roles dinámicos que intervienen en el proceso de inferencia.

```
METODO diagnóstico-basado-en-modelo-de-componentes
  DATOS: observaciones
  RESULTADOS: componentes-averiados

ALGORITMO
1.  estado := ϕ
2.  detectar(observaciones, estado -> conflictos, estado)
3.  generar(conflictos -> candidatos)
4.  WHILE |candidatos| > 1
5.    discriminar(candidatos, estado -> observación)
6     observaciones = observaciones U {observación}
7.    detectar(observaciones, estado -> conflictos, estado)
8.    generar(conflictos -> candidatos)
9.  GET-FIRST(componentes-averiados, candidatos)
```

Figura 4.13: Ejemplo de algoritmo del método de diagnóstico basado en modelo de componentes.

La figura 4.14 muestra el gráfico de la estructura global del método y la 4.15 el diagrama correspondiente a la estructura de inferencia. En el gráfico de la estructura global se ha incluido la asociación del paso de inferencia discriminar con las bases de conocimiento de probabilidad de fallos y preferencia de adquisición dado que, aunque no se ha seguido exactamente así en el ejemplo de algoritmo aquí mostrado con el fin de simplificar la descripción del método, en general estas bases se relacionan de esta forma con dicho paso de inferencia.

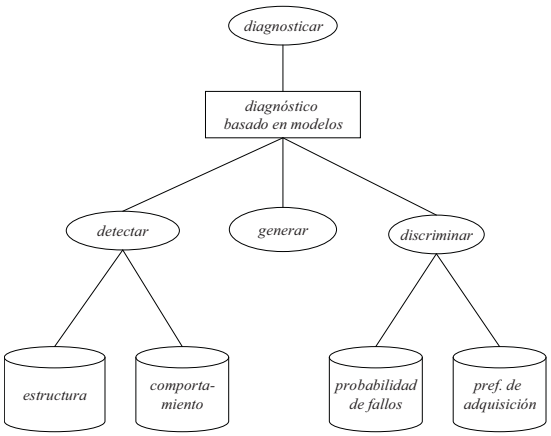


Figura 4.14: Estructura del método de diagnóstico basado en modelo de componentes.

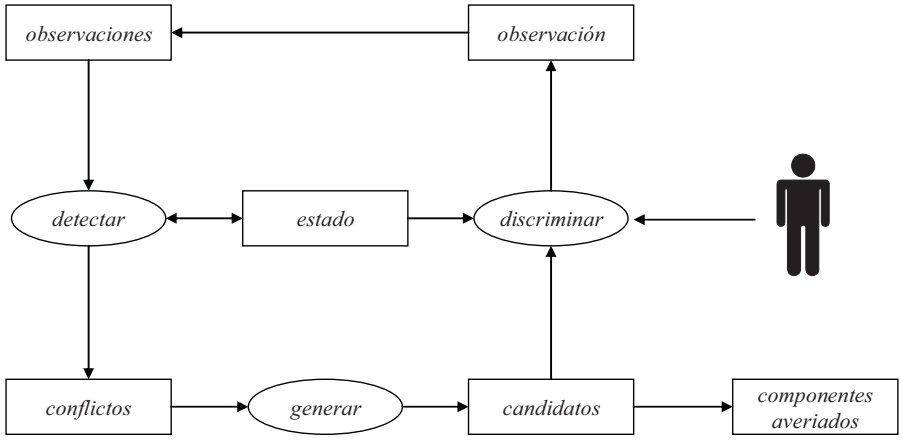


Figura 4.15: Estructura del inferencia del ejemplo de algoritmo de diagnóstico basado en modelo de componentes..

4.4 Ejemplos

4.4.1. Ejemplo 1: Ejemplo de aproximación

Considérese el siguiente ejemplo abstracto y simplificado que tiene como síntomas S1, S2, ..., S7 y causas C1, C2, ..., C8. La base de conocimiento de relaciones efectos-causa está representada con las siguientes reglas:

```
E1: S1 → C1 o C2
E2: S2 → C2 o C3 o C4
E3: S3 → C4
E4: S4 → C3
E5: S5 → C4 o C5
E6: S6 → C6
E7: S7 → C7
E8: C2 → C3 o C6 o C7
E9: C4 → C7 o C8
```

La base de conocimiento de relaciones causa-efecto está representada con el siguiente conjunto reglas:

```
M1: C3 → S4
M2: C4 → S5
M3: C6 → S6
M4: C7 → S7
```

La base de conocimiento de conocimiento circunstancial incluye las siguientes afirmaciones expresadas con el predicado *PREFERIBLE(A,B)* que indica que la causa A es preferible a B:

```
C1: PREFERIBLE(C4,C5)
C2: PREFERIBLE(C3,C2)
C3: PREFERIBLE(C3,C1)
C4: PREFERIBLE(C3,C4)
```

En un determinado momento se tienen como síntomas iniciales observados {S1, S2, S3}. Además, en caso del que el sistema durante la resolución del problema pregunte al usuario, debe responderse que: S4 es falso, S5 es verdadero, S6 es falso y S7 es verdadero.

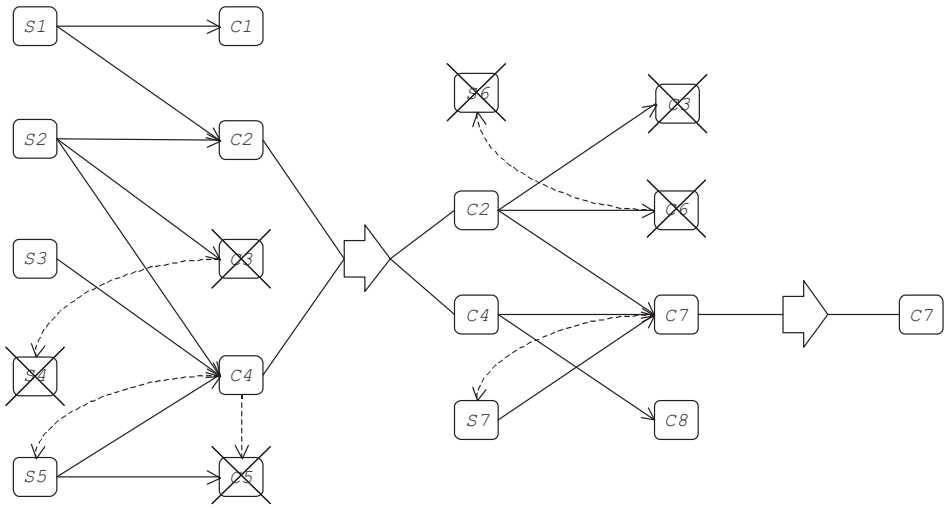


Figura 4.16: Gráfico resumen del proceso de búsqueda realizado.

Seguidamente se muestra la aplicación del algoritmo indicando cada uno de las llamadas a los pasos de inferencia y la evolución de los conjuntos considerados:

```
observaciones = {S1, S2, S3}
combinaciones = {}
eventos = {S1, S2, S3}
1. cubrir(eventos, H)
   S1 evoca C1, C2
   S2 evoca C2, C3, C4
   S3 evoca C4
   hipótesis-de-causas = {C1, C2, C3, C4}
2. diferenciar(hipótesis-de-causas, observaciones -> hipótesis-de-causas, eventos)
   C3 manifiesta S4
   PREGUNTA-SISTEMA: ¿S4?
   RESPUESTA-USUARIO: FALSO
   se rechaza C3
   C4 manifiesta S5
   PREGUNTA-SISTEMA: ¿S5?
   RESPUESTA-USUARIO: VERDADERO
   se mantiene C4
   hipótesis-de-causas = {C1, C2, C4}
   en la base de conocimiento circunstancial no
   hay sentencias que afecten a las hipótesis
   eventos = {S5}
   observaciones = {S1, S2, S3, no(S4), S5}
3. cubrir(eventos, H)
   S5 evoca C5 (además de C4)
   hipótesis-de-causas = {C1, C2, C4, C5}
4. diferenciar(hipótesis-de-causas, observaciones -> hipótesis-de-causas, eventos)
   C5 se rechaza frente a C4, aplicando conocimiento circunstancial
```

```
   hipótesis-de-causas = {C1, C2, C4}

   eventos = {}

5. combinar(hipótesis-de-causas, observaciones -> combinaciones)

   S1, S2, S3, S5 se pueden explicar con C2 y C4.

   También se pueden explicar con C1 y C4

   combinaciones = {{C2, C4}, {C1, C4}}

   eventos = {C2, C4}

6. cubrir(eventos, H)

   C2 evoca C3, C6, C7

   C4 evoca C7, C8

   hipótesis-de-causas = = {C3, C6, C7, C8}

7. diferenciar(hipótesis-de-causas, observaciones -> hipótesis-de-causas, eventos)

   C3 manifiesta S4, C3 se descarta porque no se da S4 (preguntado previamente)

   C6 manifiesta S6

   PREGUNTA-SISTEMA: ¿S6?

   RESPUESTA-USUARIO: FALSO

   se rechaza C6

   C7 manifiesta S7

   PREGUNTA-SISTEMA: ¿S7?

   RESPUESTA-USUARIO: VERDADERO

   se mantiene C7

   hipótesis-de-causas = = {C7, C8}

   en la base de conocimiento circunstancial no

   hay sentencias que afecten a las hipótesis

   eventos = {S7}

   observaciones = {S1, S2, S3, no(S4), S5, no(S6), S7}

8. cubrir(eventos, H)

   S7 sólo lo cubre C7

   hipótesis-de-causas = = {C7, C8}

9. diferenciar(hipótesis-de-causas, observaciones -> hipótesis-de-causas, eventos)
```

```
   eventos = {}
10.combinar(hipótesis-de-causas, observaciones -> combinaciones)
   S1, S2, S3, S5 se pueden explicar con C2 y C4
   C2 y C4 los explica C7 que también explica S7, por lo que una opción es {C7}
   La opción {C7, C8} se descarta por el principio de parquedad
   combinaciones = {{C7}}
   eventos = {C7}
   causas-finales = {{C7}}
   ... el proceso continúa con el otro caso de {C1, C4} (ver paso 5)
```

Por tanto una solución del problema es que la causa C7 explica los síntomas observados. La figura 4.16 muestra el gráfico resumen del proceso de búsqueda desarrollado. De forma resumida, la secuencia de preguntas-respuestas entre sistema y el usuario es la siguiente:

```
   PREGUNTA-SISTEMA: ¿S4?
   RESPUESTA-USUARIO: FALSO
   PREGUNTA-SISTEMA: ¿S5?
   RESPUESTA-USUARIO: VERDADERO
   PREGUNTA-SISTEMA: ¿S6?
   RESPUESTA-USUARIO: FALSO
   PREGUNTA-SISTEMA: ¿S7?
   RESPUESTA-USUARIO: VERDADERO
```

4.4.2. Ejemplo 2: Diagnóstico en una planta de energía térmica

El sistema MOLE se aplicó al dominio de diagnóstico de averías en una planta de energía térmica. En dicho dominio el componente principal sobre el que se hace diagnóstico es la instalación de la caldera, la unidad central de la planta. Los

problemas normalmente no hacen detener el proceso pero son una fuente de pérdidas importantes de eficiencia. Ello puede traducirse en una pérdida económica importante por consumo de combustible en exceso además del aumento significativo del envío de contaminantes a la atmósfera. El modelo que se presenta aquí es una versión simplificada del modelo construído en este problema que tiene como fin ilustrar la forma de operación del método sobre un dominio concreto. Se tiene el siguiente conocimiento relativo a relaciones efecto-causa (relaciones de tipo *evoca*):

```
E1: flujo(ceniza) = presente
    → flujo(partículas) = presente o nivel(transferencia-calorífica) = baja
E2: lectura(oxígeno)= baja
    → nivel(corriente-de-aire) = excesivo o  nivel(energía) = bajo
E3: lectura(presión)= baja
    → nivel(energía) = bajo
E4: tamaño(llama) = reducida
    → nivel(radiación) = baja
E5: nivel(corriente-de-aire) = excesivo
    → potencia(ventilación) = excesiva o nivel(material-de-mezcla) = insuficiente
E6: estado(conducción-gas) = fuga
    → nivel(transferencia-calorífica) = baja o nivel(corriente-de-aire) = excesivo
E7: nivel(transferencia-calorífica) = baja
    → nivel(radiación)= baja o flujo(corriente-de-aire) = desequilibrado
E8: flujo(corriente-de-aire) = desequilibrado
    → estado(depósito) = sucio
E9: potencia(ventilación) = excesiva
    → estado(depósito) = sucio o funcionamiento(válvula) = defectuoso
E10: velocidad(pulverizado) = lenta
    → funcionamiento(pulverización) = defectuoso o
       estado(conducto-principal) = fuga
```

Se dispone también de conocimiento sobre los síntomas que necesariamente manifiestan ciertas causas (relaciones de tipo *manifiesta*):

```
M1: nivel(corriente-de-aire)= excesivo
    → estado(conducción-gas) = fuga y lectura(oxígeno)= baja
M2: flujo(partículas)= presente
    → flujo(ceniza) = presente
M3: nivel(energía)= bajo
    → lectura(oxígeno) = baja y lectura(presión) = baja
M4: nivel(radiación)= baja
    → tamaño(llama) = reducida
M5: estado(conducto-principal)= fuga
    → velocidad(pulverizado)= lenta
M6: funcionamiento(pulverización)= defectuoso
    → velocidad(pulverizado)= lenta
```

Por último, se tienen las siguientes sentencias sobre conocimiento circunstancial para elegir entre hipótesis (`PREFERIBLE(A,B)` indica que la hipótesis de causa A es preferible a la hipótesis de causa B). :

```
C1: PREFERIBLE(potencia(ventilación)= excesiva,
               nivel(material-de-mezcla)= insuficiente)
C2: PREFERIBLE(funcionamiento(válvula)= defectuoso,
               funcionamiento(pulverización)= defectuoso)
C3: PREFERIBLE(estado(conducto-principal)= fuga,
               funcionamiento(pulverización)= defectuoso)
C4: PREFERIBLE(nivel(corriente-de-aire)= excesivo,
               estado(conducto-principal)= fuga)
```

Teniendo en cuenta que en un cierto momento se registran como síntomas `flujo(ceniza) = presente` *y* `estado(conducción-gas) = fuga`, se aplica el método de *cubrir-y-diferenciar* para determinar la posible avería. A continuación se indican la evolución del proceso de resolución indicando las llamadas a los diferentes pasos de inferencia. La figura 4.17 resume la búsqueda realizada.

observaciones = {flujo(ceniza)=presente,estado(conducción-gas)=fuga}

combinaciones = {}

eventos = {flujo(ceniza)=presente,estado(conducción-gas)=fuga}

1. cubrir(eventos, H)

   REGLAS:

   E1: flujo(ceniza) = presente

   → flujo(partículas) = presente o nivel(transferencia-calorífica) = baja

   E6: estado(conducción-gas) = fuga

   → nivel(transferencia-calorífica) = baja o nivel(corriente-de-aire) = excesivo

   hipótesis-de-causas = {flujo(partículas)=presente,

   nivel(transferencia-calorífica)=baja, nivel(corriente-de-aire)=excesivo}

2. diferenciar(hipótesis-de-causas, observaciones -> hipótesis-de-causas, eventos)

   REGLAS:

   M1: nivel(corriente-de-aire)= excesivo

   → estado(conducción-gas) = fuga y lectura(oxígeno)= baja

   PREGUNTA-SISTEMA: ¿lectura(oxígeno)?

   RESPUESTA-USUARIO: baja

   se mantiene nivel(corriente-de-aire)= excesivo

   en la base de conocimiento circunstancial no

   hay sentencias que afecten a las hipótesis

   hipótesis-de-causas = {flujo(partículas)=presente,

   nivel(transferencia-calorífica)=baja, nivel(corriente-de-aire)=excesivo}

   eventos = {lectura(oxígeno)=baja}

   observaciones = {lectura(oxígeno)=baja, flujo(ceniza)=presente,

   estado(conducción-gas)=fuga}

3. cubrir(eventos, H)

   REGLAS:

   E2: lectura(oxígeno)= baja

   → nivel(corriente-de-aire) = excesivo o nivel(energía) = bajo

hipótesis-de-causas = {flujo(partículas)=presente,

nivel(transferencia-calorífica)=baja, nivel(corriente-de-aire)=excesivo,

nivel(energía) = bajo}

4. diferenciar(hipótesis-de-causas, observaciones -> hipótesis-de-causas, eventos)

REGLAS:

M3: nivel(energía)= bajo

→ lectura(oxígeno) = baja y lectura(presión) = baja

(no se muestran reglas utilizadas que dan resultados ya obtenidos)

PREGUNTA-SISTEMA: ¿lectura(presión)?

RESPUESTA-USUARIO: normal

se descarta nivel(energía)= bajo

en la base de conocimiento circunstancial no

hay sentencias que afecten a las hipótesis

hipótesis-de-causas = {flujo(partículas)=presente,

nivel(transferencia-calorífica)=baja, nivel(corriente-de-aire)=excesivo}

eventos = {lectura(presión)=normal}

observaciones = {lectura(presión)=normal, lectura(oxígeno)=baja,

flujo(ceniza)=presente, estado(conducción-gas)=fuga}

5. cubrir(eventos, H)

no hay reglas para lectura(presión)=normal

hipótesis-de-causas = {flujo(partículas)=presente,

nivel(transferencia-calorífica)=baja, nivel(corriente-de-aire)=excesivo}

6. diferenciar(hipótesis-de-causas, observaciones -> hipótesis-de-causas, eventos)

no hay nuevas reglas para diferenciación en la base de causas-efectos

ni en la de conocimiento circunstancial

hipótesis-de-causas = {flujo(partículas)=presente,

nivel(transferencia-calorífica)=baja, nivel(corriente-de-aire)=excesivo}

eventos = {}

observaciones = {lectura(presión)=normal, lectura(oxígeno)=baja,

flujo(ceniza)=presente, estado(conducción-gas)=fuga}

```
7. combinar(hipótesis-de-causas, observaciones -> combinaciones)
   combinaciones =
      {{nivel(transferencia-calorífica)=baja, nivel(corriente-de-aire)=excesivo},
       {flujo(partículas)=presente, nivel(corriente-de-aire)=excesivo}}
   eventos =
      {{nivel(transferencia-calorífica)=baja, nivel(corriente-de-aire)=excesivo}
8. cubrir(eventos, H)
   REGLAS:
   E5: nivel(corriente-de-aire) = excesivo
     → potencia(ventilación) = excesiva o nivel(material-de-mezcla) = insuficiente
   E7: nivel(transferencia-calorífica) = baja
     → nivel(radiación)= baja o flujo(corriente-de-aire) = desequilibrado
   hipótesis-de-causas = {potencia(ventilación)=excesiva,
      nivel(material-de-mezcla)=insuficiente, nivel(radiación)=baja,
      flujo(corriente-de-aire)=desequilibrado}
9. diferenciar(hipótesis-de-causas, observaciones -> hipótesis-de-causas, eventos)
   REGLAS:
   M4: nivel(radiación)= baja → tamaño(llama) = reducida
   PREGUNTA-SISTEMA: ¿tamaño(llama)?
   RESPUESTA-USUARIO: normal
   se descarta nivel(radiación)=baja
   SENTENCIAS:(de la base de conocimiento circunstancial)
   C1: PREFERIBLE(potencia(ventilación)=excesiva,
                  nivel(material-de-mezcla)=insuficiente)
   se descarta nivel(material-de-mezcla)=insuficiente
   hipótesis-de-causas = {potencia(ventilación) = excesiva,
                          flujo(corriente-de-aire) = desequilibrado}
   eventos = {tamaño(llama)=normal}
   observaciones = {tamaño(llama)=normal, lectura(presión)=normal,
                    lectura(oxígeno)=baja, flujo(ceniza)=presente,
                  estado(conducción-gas)=fuga}
```

```
10.cubrir(eventos, H)
   no hay reglas para tamaño(llama)=normal
   hipótesis-de-causas = {potencia(ventilación) = excesiva,
                          flujo(corriente-de-aire) = desequilibrado}
11.diferenciar(hipótesis-de-causas, observaciones -> hipótesis-de-causas, eventos)
   no hay nuevas reglas para diferenciación en la base de causas-efectos
   ni en la de conocimiento circunstancial
   hipótesis-de-causas = {potencia(ventilación) = excesiva,
                          flujo(corriente-de-aire) = desequilibrado}
   eventos = {}
   observaciones = {tamaño(llama)=normal, lectura(presión)=normal,
                    lectura(oxígeno)=baja, flujo(ceniza)=presente,
                  estado(conducción-gas)=fuga}
12.combinar(hipótesis-de-causas, observaciones -> combinaciones)
   combinaciones =
      {{potencia(ventilación)=excesiva, flujo(corriente-de-aire)=desequilibrado}}
   eventos =
      {potencia(ventilación)=excesiva, flujo(corriente-de-aire)=desequilibrado}
13. cubrir(eventos, H)
   REGLAS:
   E8: flujo(corriente-de-aire) = desequilibrado
     → estado(depósito) = sucio
   E9: potencia(ventilación) = excesiva
     → estado(depósito) = sucio o funcionamiento(válvula) = defectuoso
   hipótesis-de-causas =
       {estado(depósito)=sucio, funcionamiento(válvula)=defectuoso}
14.diferenciar(hipótesis-de-causas, observaciones -> hipótesis-de-causas, eventos)
   no hay reglas para diferenciación en la base de causas-efectos
   ni en la de conocimiento circunstancial
   hipótesis-de-causas =
      {estado(depósito)=sucio, funcionamiento(válvula)=defectuoso}
```

```
   eventos = {}
   observaciones = {tamaño(llama)=normal, lectura(presión)=normal,
                    lectura(oxígeno)=baja, flujo(ceniza)=presente,
                  estado(conducción-gas)=fuga}
15.combinar(hipótesis-de-causas, observaciones -> combinaciones)
   combinaciones =
      {{estado(depósito)=sucio}}
   eventos =
      {estado(depósito)=sucio}
   causas-finales = {{estado(depósito)=sucio}}
   ... el proceso continúa con la alternativa del paso 7 ...
```

Por lo tanto, una solución del problema que explica todos los síntomas y observaciones es:

```
   estado(depósito)=sucio
```

La interacción entre usuario y sistema ha sido la siguiente:

```
   PREGUNTA-SISTEMA: ¿lectura(oxígeno)?
   RESPUESTA-USUARIO: baja
   PREGUNTA-SISTEMA: ¿lectura(presión)?
   RESPUESTA-USUARIO: normal
   PREGUNTA-SISTEMA: ¿tamaño(llama)?
   RESPUESTA-USUARIO: normal
```

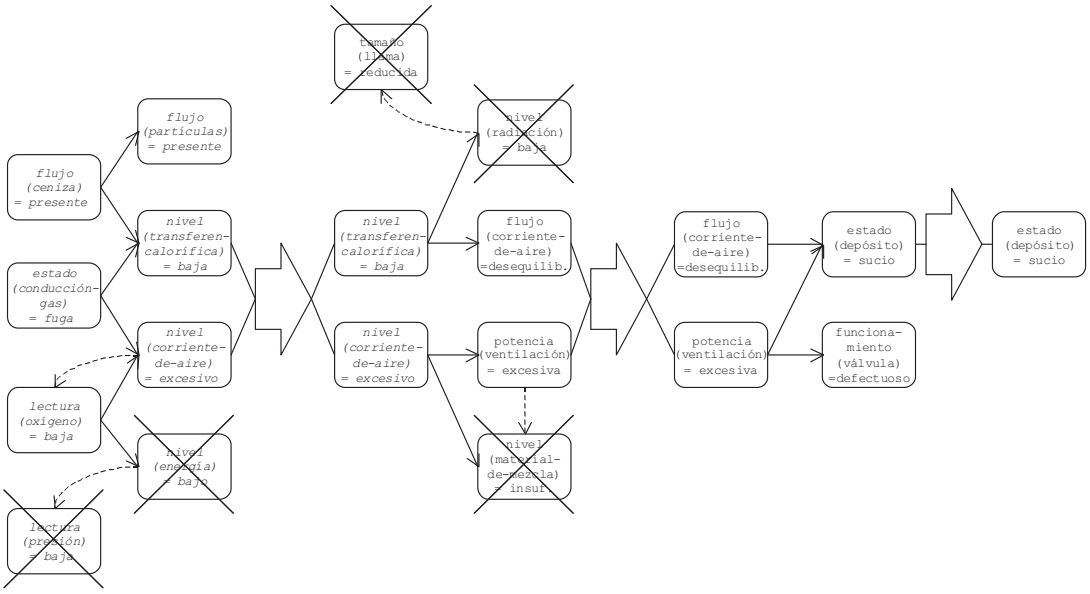


Figura 4.17: Resumen de la búsqueda realizada en el proceso.

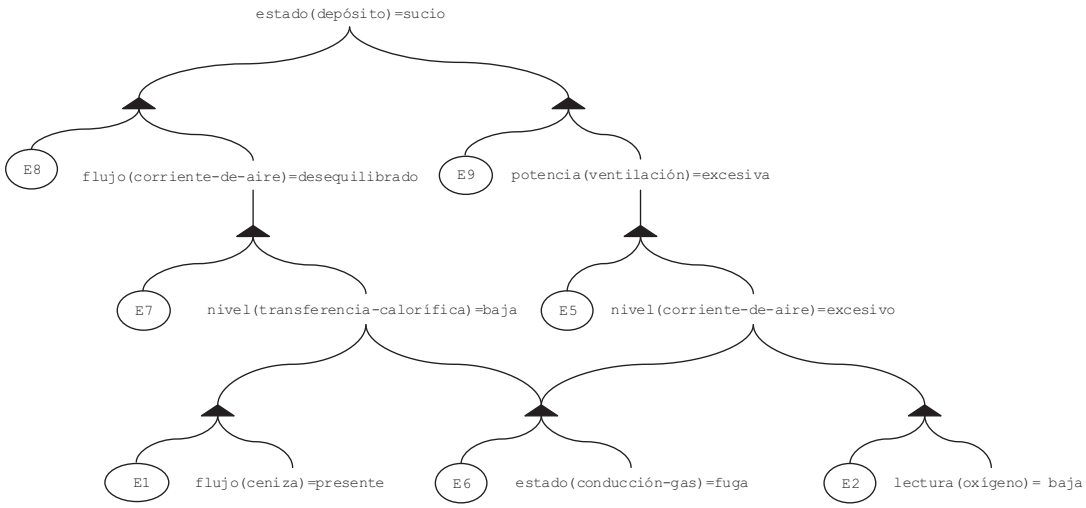


Figura 4.18: Árbol explicativo de la conclusión alcanzada.

4.4.3. Ejemplo 3: Diagnóstico en un circuito digital

Se desarrolla aquí paso a paso el ejemplo del circuito utilizado en la descripción del método de diagnóstico basado en modelo de componentes. Los detalles de operación sobre la construcción del estado ATMS pueden consultarse en el anexo en el presente libro. El problema parte de una situación en la que se dispone de las medidas sobre las variables A, B, C, D, E y F.

```
observaciones = {A=3,B=2,C=2,D=3,E=3,F=10}

estado = {}

1. detectar(observaciones, estado -> conflictos, estado)

   estado = {<A=3,{}>, <B=2,{}>,<C=2,{}>, <D=3,{}>, <E=3,{}>,

             <X=4,{{A1,M2}}>, <X=6,{{M1}}>, <Y=4,{{A1,M1}}>, <Y=6,{{M2}}>,

             <Z=6,{{M3}}>, <F=10,{}>, <G=10,{{A1,A2,M1,M3}}, <G=12,{{A2,M2,M3}}>,

             <no-good,{{A1,M1,M2}}>}

   conflictos = {{A1,M1,M2}}

2. generar(conflictos -> candidatos)

   los candidatos mínimos son {{A1},{M1},{M2}}

   los candidatos de cardinalidad mínima son {{A1},{M1},{M2}}

   candidatos = {{A1},{M1},{M2}}

3. discriminar(candidatos, estado -> observación)

       G=10, S={{M2}}

       G=12, S={{A1},{M1}}

     P(G) = 1 ln 1 + 2 ln 2 = 1.4

       X=4, S={{M1}}

       X=6, S={{A1},{M2}}

     P(X) = 1 ln 1 + 2 ln 2 = 1.4

       Y=4, S={{M2}}

       Y=6, S={{A1},{M1}}
```

```
     P(Y) = 1 ln 1 + 2 ln 2 = 1.4

       Z=8, S={{A1},{M1},{M2}}

     P(Z) = 3 ln 3 = 3.29

   Son igualmente preferibles G, X e Y. Se elige G.

   PREGUNTA-SISTEMA: ¿valor de la medida G?

   RESPUESTA-USUARIO: G=12

   observación = G=12

   observaciones = {A=3,B=2,C=2,D=3,E=3,F=10,G=12}

4. detectar(observaciones, estado -> conflictos, estado)

   estado = {<A=3,{}>, <B=2,{}>,<C=2,{}>, <D=3,{}>, <E=3,{}>,
             <X=4,{{A1,M2},{A1,A2,M3}}>, <X=6,{{M1}}>,
             <Y=4,{{A1,M1}}>, <Y=6,{{M2},{A2,M3}}>,
             <Z=6,{{M3},{A2,M2}}>, <Z=8,{{A1,A2,M1}}>, <F=10,{}>, <G=12,{}>,
             <no-good,{{A1,M1,M2},{A1,A2,M1,M3}}>}

   conflictos = {{A1,M1,M2},{A1,A2,M1,M3}}

5. generar(conflictos -> candidatos)

   los candidatos mínimos son {{A1},{M1},{M2,A2},{M2,M3}}

   los candidatos de cardinalidad mínima son {{A1},{M1}}

   candidatos = {{A1},{M1}}

6. discriminar(candidatos, estado -> observación)

       X=4, S={{M1}}

       X=6, S={{A1}}

     P(X) = 1 ln 1 + 1 ln 1 = 0

       Y=6, S={{A1},{M1}}

     P(Y) = 2 ln 2 = 1.4

       Z=6, S={{A1},{M1}}

     P(Z) = 2 ln 2 = 1.4

   El mínimo valor lo tiene X

   PREGUNTA-SISTEMA: ¿valor de la medida X?

   RESPUESTA-USUARIO: X=6
```

```
   observación = X=6

   observaciones = {A=3,B=2,C=2,D=3,E=3,F=10,G=12,X=6}

7. detectar(observaciones, estado -> conflictos, estado)

   estado = {<A=3,{}>, <B=2,{}>,<C=2,{}>, <D=3,{}>, <E=3,{}>,<X=6,{}>,

             <Y=4,{{A1,M1}}>, <Y=6,{{M2},{A2,M3}}>,

             <Z=6,{{M3},{A2,M2}}>, <Z=8,{{A1,A2,M1}}>, <F=10,{}>, <G=12,{}>,

             <no-good,{{A1,M2},{A1,A2,M3}}>}

   conflictos = {{A1,M2},{A1,A2,M3}}

8. generar(conflictos -> candidatos)

   los candidatos mínimos son {{A1},{A2,M2},{M2,M3}}

   los candidatos de cardinalidad mínima son {{A1}}

   candidatos = {{A1}}

|candidatos| = 1 => terminación del bucle

FIN con el resultado: componentes-averiados = {A1}
```

Por tanto, la solución final es que falla el componente A1. Durante el proceso de diagnóstico el sistema ha realizado la siguiente interacción con el usuario para preguntar el valor de las medidas G y X:

```
   PREGUNTA-SISTEMA: ¿valor de la medida G?
   RESPUESTA-USUARIO: G=12
   PREGUNTA-SISTEMA: ¿valor de la medida X?
   RESPUESTA-USUARIO: X=6
```

4.4.4. Ejemplo 4: Diagnóstico en circuito con probabilidades de fallo

En este apartado se muestra un ejemplo de diagnóstico descrito por [Stefik, 95] que tiene como fin ilustrar la forma original que plantea el método GDE para dirigir el proceso de obtención de medidas para realizar el diagnóstico haciendo uso de probabilidades de fallo de componentes. El ejemplo ilustra también el manejo de modelos de fallo. Además, al final de la descripción se incluye la forma de

simplificar el proceso asumiendo ciertas hipótesis, dando lugar a la formulación que ha sido referenciada en el método general.

Considérese el circuito de la figura 4.19 en donde M1 y M2 son multiplicadores y A1 y A2 son sumadores. La precisión está limitada a 5 bits, por lo que los valores están entre 0 y 31. Si las entradas al sumador son X e Y, la salida será mod(X + Y, 32) en donde mod(A, B) es la operación módulo que devuelve el resto de dividir A entre B. La salida del multiplicador será mod(X * Y, 32).

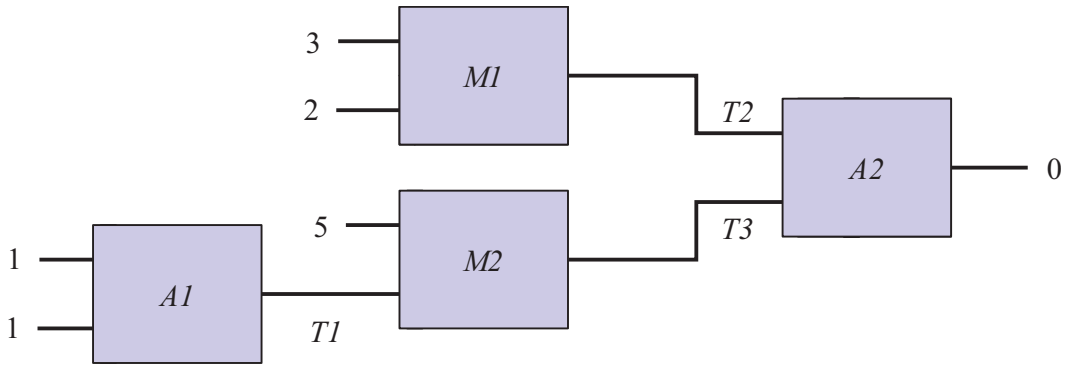


Figura 4.19: Ejemplo de circuito para diagnóstico [Stefik, 95].

Cada uno de estos componentes puede adoptar cuatro posibles estados {*normal, cortado, fallo-bit, desconocido}* que se refieren respectivamente a (1) operación normal bajo un funcionamiento correcto, (2) circuito interior cortado, (3) fallo en el bit más significativo y (4) desconocido. En la avería correspondiente al estado *cortado* la salida es siempre 0. Cuando el estado es *fallo-bit*, la salida pierde el valor del bit más significativo lo que supone que las operaciones se expresarán sólo en 4 bits. En este caso la suma será mod(X + Y, 16) y la multiplicación mod(X * Y, 16). En el estado desconocido la salida será desconocida.

Las probabilidades de fallo dadas por el fabricante de componentes electrónicos para un multiplicador M y para un sumador A son:

```
P(estado(M, normal)) =  0.9990
P(estado(M, cortado)) =  0.0003
P(estado(M, fallo-bit)) =  0.0006
P(estado(M, desconocido)) = 0.0001
P(estado(A, normal)) =  0.9990
P(estado(A, cortado)) =  0.0003
P(estado(A, fallo-bit)) =  0.0006
P(estado(A, desconocido)) = 0.0001
```

Durante el proceso de diagnóstico, los candidatos indican las posibles averías. Un candidato se expresa como un conjunto de estados de componentes averiados asumiendo que el resto de los componentes están en estado normal. Por ejemplo, el candidato {estado(A2, fallo-bit)} expresa que A2 está en el modo de avería fallo-bit y los demás componentes operan correctamente. El candidato {estado(A2, cortado)} expresa que A2 está en el modo de avería cortado y los demás componentes funcionan bien. El candidato {estado(M1,cortado), estado(M2, cortado)} expresa que los componentes M1 y M2 están en el estado de avería cortado mientras que el resto operan correctamente.

Estos tres candidatos se han obtenido tras un proceso de análisis de las medidas registradas, centrándose en candidatos con uno o dos componentes averiados (candidatos de más componentes tienen muy baja probabilidad). También se han descartado candidatos que incluyen modo desconocido debido a su baja probabilidad en comparación con los otros candidatos.

Las probabilidades a priori de dichos candidatos se calculan multiplicando las probabilidades a priori de los componentes según su estado:

P({estado(A2,fallo-bit)}) = P (estado(A1, normal)) * P(estado(A2, fallo-bit) *
* P (estado(M1, normal)) * P (estado(M1,normal)) =
= 0.9990 * 0.0006 * 0.9990 * 0.9990 = $5.982\ 10^{-3}$

P({estado(A2,cortado)}) = P (estado(A1, normal)) * P(estado(A2, cortado) *
* P (estado(M1, normal)) * P (estado(M1,normal)) =
= 0.9990 * 0.0003 * 0.9990 * 0.9990 = $2.991\ 10^{-3}$

P({estado(M1,cortado), estado(M1,cortado)}) = P (estado(A1, normal)) *
* P(estado(A2, normal) * P (estado(M1, cortado)) *
* P (estado(M1, cortado)) =
= 0.9990 * 0.9990 * 0.0003 * 0.0003 = $8.982\ 10^{-8}$

La forma de decidir la siguiente medición a realizar está basada en la medida de entropía de Shannon. Si P_i son las probabilidades de los candidatos. La medida de entropía se calcula de acuerdo con la fórmula:

$$H = - \sum P_i \ln P_i$$

Teniendo en cuenta que en el ejemplo sólo son válidos los tres candidatos anteriores, sus probabilidades $5.982\ 10^{-3}$, $2.991\ 10^{-3}$ y $8.982\ 10^{-8}$ se normalizan para que sumen la unidad manteniendo la misma proporción (dividiendo cada una por la suma de todas ellas) lo que da lugar respectivamente a las probabilidades de: 0.66666, 0.33333 y 0.00001. El valor de H para este caso es:

H = - 0.66666 ln(0.66666) - 0.33333 ln(0.33333) - 0.0000.1 ln(0.0000.1) =
= 0.2703 + 0.3662 + 0.00011 = 0.6366

Los valores de H cercanos a cero indican una distribución de probabilidad menos plana y menos distribuida uniformemente por los candidatos, por lo que el diagnóstico será más afinado en situaciones con H cercano a cero. Para decidir qué medición es mejor, se busca aquélla que previsiblemente produzca una mejor

redistribución de las probabilidades dando lugar a un menor valor de H. Esto se realiza de acuerdo con la estrategia denominada *one step look ahead*, ya mencionada previamente. Según esta estrategia se analiza el previsible efecto en la disminución de la entropía que supone una única medición, sin tener en cuenta qué mediciones se hagan después.

Supóngase un caso sobre el circuito considerado en donde la salida medida de A2 es 6 (en vez del valor 0 que mostraba la figura 4.19). Dado que la salida esperada es 16 en vez de 6, alguno de los componentes debe estar averiado (uno o varios). En este caso, razonando sobre el modelo en sentido directo e inverso se tienen las siguientes conclusiones similares a las que se obtienen con GDE en forma de estado de ATMS sobre el funcionamiento correcto de componentes:

```
T1 = 2,  si {estado(A1,normal)}
T2 = 0,  si {estado(M1,normal), estado(M2,normal),estado(A2,normal)}
T2 = 6,  si {estado(M1,normal)}
T3 = 10, si {estado(A1,normal), estado(M2,normal)}
T3 = 3,  si {estado(A2,normal), estado(M1,normal)}
```

Los siguientes candidatos de un solo elemento explican las medidas observadas:

```
{estado(A1,cortado)}
{estado(M2,cortado)}
{estado(A1,desconocido)}
{estado(A2,desconocido)}
{estado(M1,desconocido)}
{estado(M2,desconocido)}
```

Las medidas de probabilidad de dichos candidatos son:

$P(\{estado(A1,cortado)\}) = 2.991\ 10^{-4}$

$P(\{estado(M2,cortado)\}) = 2.991\ 10^{-4}$

$P(\{estado(A1,desconocido)\}) = 9.970\ 10^{-5}$

$P(\{estado(A2,desconocido)\}) = 9.970\ 10^{-5}$

$P(\{estado(M1,desconocido)\}) = 9.970\ 10^{-5}$

$P(\{estado(M2,desconocido)\}) = 9.970\ 10^{-5}$

También hay otros candidatos de más de un elemento que pueden explicar las medidas. Por ejemplo el candidato {estado(A1, cortado), estado(M2, cortado)} tiene una probabilidad de fallo muy baja de $8.982\ 10^{-8}$. Todos los candidatos de cardinalidad 2 y superior tienen una probabilidad muy inferior a los candidatos considerados de cardinalidad 1 por lo que desde un punto de vista práctico se pueden descartar. Las probabilidades de los candidatos considerados se normalizan dividiendo por la suma de todos ellos con el fin de que sumen la unidad. Esta normalización es importante con el fin de permitir una adecuada interpretación y comparación de valores de entropía. Esto da lugar a los siguientes valores:

$P(\{estado(A1,cortado)\}) = 0.3$

$P(\{estado(M2,cortado)\}) = 0.3$

$P(\{estado(A1,desconocido)\}) = 0.1$

$P(\{estado(A2,desconocido)\}) = 0.1$

$P(\{estado(M1,desconocido)\}) = 0.1$

$P(\{estado(M2,desconocido)\}) = 0.1$

Entre estos candidatos, cuando se estudia la posibilidad de una medición y se tiene en cuenta el valor esperado, los candidatos correspondientes a estados desconocidos no se consideran a no ser que no haya explicaciones posibles dadas por otros modos

de comportamiento. En general, cuando considera un candidato C_j con respecto a una medición realizada $T = V_i$ se pueden dar varias situaciones:

1) C_j es incompatible con la medida observada dado que predice un valor $T = V_j$ en donde V_j es diferente de V_i, por lo que el candidato se descarta para dicha situación.

2) C_j predice el mismo valor medido, entonces se tiene en cuenta para una posterior discriminación, incluyéndolo en la lista de posibles candidatos en donde se realiza una normalización de la probabilidad.

3) C_j no predice ningún valor que se pueda medir (se denomina candidato no comprometido), por lo que no se puede descartar ni confirmar haciendo alguna medición (en este caso se puede asociar como candidato posible a cada medición pendiente dividiendo su probabilidad entre el número de valores posibles de dichas mediciones).

4) C_j tiene un modo desconocido que no permite predecir el valor de la variable (alguno de los componentes del candidato está en modo desconocido), en este caso puede considerarse como no comprometido, o bien, tal como se hace en este ejemplo, sencillamente no se considera cuando no se puede distinguir su comportamiento de otro modo de operación.

Para elegir la medición más informativa se actúa de la siguiente forma. Se consideran las medidas T sobre las que se tienen varios valores posibles V_i. Para cada valor V_i hay un conjunto de candidatos consistentes con V_i. Se obtiene $H(T = V_i)$ que cuantifica la entropía que resultaría si se mide $T = V_i$. Para calcular $H(T = V_i)$ se consideran los candidatos que son consistentes con $T = V_i$.

En el ejemplo considerado, de las tres medidas posibles T1, T2 y T3, hay más de una alternativa para T1 y T3. Para T2 se espera medir sólo una única medida, T2 = 6, por lo que no es necesario considerarla en el proceso de decisión sobre dónde hacer la medición.

Para la medida T1 se tienen dos valores posibles T1=2 y T1=0. Con T1=2, sólo el candidato {estado(M2, cortado)} es consistente con ese valor. El cálculo de la entropía tiene en cuenta la probabilidad de dicho candidato debidamente normalizada teniendo en cuenta todos los candidatos posibles. Así, la normalización de p({estado(M2, cortado)} = 0.3 da lugar a p({estado(M2, cortado)} = 1.0 dado que hay un único candidato. La entropía para este caso es:

```
H(T1 = 2) = - p({estado(M2, cortado)} ln(p({estado(M2, cortado)}) =
          = 1.0 ln(1.0) = 0
```

Para T1 = 0, el candidato consistente es {estado(A1, cortado)}. Por razones similares al caso anterior se tiene:

```
H(T1 = 0) = - p({estado(A1, cortado)} ln(p({estado(A1, cortado)}) =
          = - 1.0 ln(1.0) = 0
```

En el caso de T3 se tienen dos posibilidades T3 = 0 y T3 = 10. Para T3 = 0 hay dos candidatos consistentes {estado(A1, cortado)} y {estado(M2, cortado)} ambos con probabilidad 0.3. El resultado de la normalización ajusta sus probabilidades a 0.5 para que la suma sea la unidad. El valor de la entropía es:

```
H(T3 = 0) = - p({estado(A1, cortado)} ln(p({estado(A1, cortado)}) -
            - p({estado(M1, cortado)} ln(p({estado(M1, cortado)}) =
          = - 0.5 ln (0.5) - 0.5 ln (0.5)= 0,6931
```

Para T3 = 10 ninguno de los dos candidatos son consistentes con la medida. Por tanto se consideran aquí candidatos correspondientes a estados desconocidos. De los posibles, sólo el caso del sumador A2 es consistente, dado que A1 y M2 deben operar correctamente para que se obtenga T3 = 10. Además el funcionamiento correcto de A2 y M1 averiado con estado desconocido no explicaría la medida de valor 6 de salida de A2 con T3 = 10. Por tanto el candidato es {estado(A2, desconocido)} con probabilidad 0.1 cuya normalización, dado que es único, da lugar a una medida de 1.0. Con ello se obtiene:

```
H(T3 = 10) = - p({estado(A2, desconocido)} ln(p({estado(A2, desconocido)}) =
           = - 1.0 ln(1.0) = 0
```

A su vez, la probabilidad $P(T = V_i)$ de que una medida T tome un determinado valor V_i se calcula como la suma de las probabilidades de todos los candidatos que son consistentes con dicha medida.

```
P(T1 = 2) = p({estado(M2, cortado)} = 0.3
P(T1 = 0) = p({estado(A1, cortado)} = 0.3
P(T3 =10) = p({estado(A2, desconocido)} = 0.1
P(T3 = 0) = p({estado(A1, cortado)} + p({estado(M1, cortado)} =
          = 0.3 + 0.3 = 0.6
```

Finalmente, la entropía esperada correspondiente a una determinada medida *T* se calcula con:

$$H_e(T) = \Sigma\, P(T = V_i)\, H(T = V_i)$$

Lo que da lugar a los siguientes valores:

$$H_e(T1) = P(T1 = 2)\ H(T1 = 2) + P(T1 = 0)\ H(T1 = 0) =$$

$$= 0.3\ 0 + 0.3\ 0 = 0$$

$$H_e(T3) = P(T3 = 0)\ H(T3 = 0) + P(T3 = 10)\ H(T3 = 10) =$$

$$= 0.6\ 0.6931 + 0.1\ 0 = 0.4159$$

La medida que obtenga menor valor de entropía esperada será la elegida, dado que se espera que distribuya de una forma más afinada (menos uniforme) la probabilidad de los candidatos restantes. En el ejemplo, la medida elegida es T1.

A partir de este ejemplo es fácil ver cómo se pueden simplificar los cálculos si, tal como se realizó en la descripción general del método, se hace la simplificación de que las probabilidades de fallo de los componentes son muy pequeñas e iguales. En ese caso $H\ (T = V_i) = -\ \Sigma P_j\ ln\ P_j$ en donde cada P_j es la probabilidad normalizada de cada candidato consistente con $T = V_i$. Todos los valores P_j son iguales y, dado que se trata de una medida normalizada, $P_j = 1/N_i$ en donde N_i es el número de candidatos consistentes con $T=V_i$. Con ello se puede reducir la expresión:

$$H(T = V_i) = -\ \Sigma\ P_j \ln P_j = -1/N_i \Sigma \ln 1/N_i = -\ \ln 1/N_i = \ln N_i$$

Además, el valor $P(T = V_i)$ se calcula como la suma de las probabilidades de los candidatos consistentes con $T=V_i$ es decir con $P(T = V_i) = \Sigma P'_j$ en donde la medida P'_j es la probabilidad de un candidato C_j que es consistente con $P(T = V_i)$. Dado que la probabilidad es igual en todos los candidatos y debe sumar la unidad, se tiene $P'_j = 1/N$ siendo N el número total de candidatos. De acuerdo con ello, la expresión se reduce a:

$$P(T = V_i) = \Sigma\ P'_j = Ni / N$$

Sustituyendo en la fórmula general:

$$H_e(T) = \sum P(T = V_i)\, H(T = V_i) = 1/N \sum N_i \ln N_i$$

Mediante eliminación del valor *N* que es independiente de *T*, se obtiene la formulación más simple presentada en la descripción general del método:

$$H_e(T) = \sum N_i \ln N_i$$

4.5 Ejercicios

EJERCICIO 4.1. Considerar el siguiente conjunto de sentencias relativas al problema de diagnóstico médico:

```
S1: Si velocidad-sedimentación-alta-en-análisis-de-sangre entonces
    el paciente puede tener infección-bacteriana
S2: Si piedra-en-vesícula-biliar entonces debe observarse patrón-
    de-obstrucción-en-rayos-X
S3: Si operación-quirúrgica-reciente entonces es más probable
    inflamación-de-conducto que tumor
S4: Si paciente-tiene-hepatitis entonces debe haber transaminasas-
    altas-en-análisis-de-sangre
S5: Si no-intervenciones-recientes entonces descartar mala-
    cicatrización
S6: Si dolor-agudo-abdominal entonces puede haber piedra-en-
    vesícula-biliar
```

SE PIDE:

Teniendo en cuenta que desea aplicar el método de resolución de problemas de *cubrir y diferenciar*, responder de forma razonada a qué clase de conocimiento corresponde cada una de las sentencias anteriores de acuerdo con los tipos de conocimiento de dicho método.

EJERCICIO 4.2. Se dispone de un modelo basado en el método *cubrir-y-diferenciar* sobre el que se tiene el siguiente conocimiento relativo a relaciones efectos-causa:

A → G	I → M, N
B → G o H	J → N
C → G o H o I	K → P o Q
D → H o I	L → P o Q
E → I	M → P o Q o R
F → L	N → P o Q o R
G → L o M	O → R
H → L o M o N	

Por ejemplo, la relación B → G o H indica que el efecto B se explica por la causa G o la causa H. Se dispone también de conocimiento sobre los síntomas que necesariamente manifiestan ciertas causas:

G → A y B	N → J
H → B y D	P → K
I → E	R → O
L → F	

En estas relaciones, por ejemplo, G → A y B significa que la causa G necesariamente tiene que manifestar los síntomas A y B.

SE PIDE:

Sabiendo que en un cierto momento se registra el síntoma B aplicar el método de *cubrir-y-diferenciar* para determinar las posibles causas. En el proceso de resolución, en caso de que sea necesario conocer su valor, considerar como verdaderos A, D, F, J, K y falsos C, E, O. Mostrar el gráfico que ilustra el proceso de resolución.

EJERCICIO 4.3. Se desea formular un modelo para diagnóstico médico sobre enfermedades relacionadas con el hígado. Para ello, se definen las siguientes variables relativas a síntomas y posibles causas:

CO: bloqueo de secreción de bilis (colestasis)
DA: dolor agudo abdominal
PF: presencia de fiebre
CP: colangiograma positivo
PP: piedra en páncreas
PV: piedra en vesícula biliar
TV: tumor en la vesícula biliar
RX: patrón de dilatación observado en rayos X
EC: estrechamiento del conducto
PD: problemas digestivos
IR: irritación del conducto
MC: mala cicatrización del conducto
CA: cicatriz en zona abdominal

Con el fin de simular el proceso de diagnóstico, se plantea utilizar el método *cubrir-y-diferenciar*. Para ello, se dispone del siguiente conocimiento relativo a relaciones efecto-causa:

PF → PV o TV	RX → EC
DA → PP o PV	PD → IR
CP → PP o PV	EC → IR o MC
CO → PP o PV o TV o EC	CA → MC

Por ejemplo, la relación PF → PV o TV indica que el efecto PF, *presencia de fiebre*, evoca la causa PV, *piedra en la vesícula biliar*, (es decir, el efecto PF permite sospechar la presencia de la causa PV). PF también evoca la causa TV, *tumor en la vesícula biliar*. Se dispone también de conocimiento sobre los síntomas que ciertas causas manifiestan:

PP → CP y CO	EC → RX
PV → CP y CO	IR → PD
TV → CO	MC → CA

En estas relaciones, por ejemplo, PP → CP y CO significa que la causa PP necesariamente tiene que manifestar los síntomas CP y CO. Por último, se dispone también de conocimiento sobre conocimiento circunstancial entre hipótesis:

PV > PP	PP > TV
PV > TV	EC > TV

En las relaciones anteriores, por ejemplo, PV > PP indica que la hipótesis de causa PV es preferible a la hipótesis de causa PP.

SE PIDE:

1. Teniendo en cuenta que en un cierto momento se tiene como único síntoma *bloqueo de secreción de bilis*, aplicar el método de *cubrir-y-diferenciar* para determinar la posible o posibles causas de dicho síntoma. En el proceso de resolución, en caso de que sea necesario conocer su valor, considerar como verdadero los hechos RX y CA y falso CP.

2. Considérese ahora un caso diferente al anterior en donde los síntomas observados son: *dolor agudo abdominal* y *bloqueo de secreción de bilis*. Durante la resolución, si es necesario conocer su valor se debe asumir como verdadero CP, DA, RX , PD y falso CA. ¿Cuál o cuáles son las posibles causas de dichos síntomas?

3. Como alternativa a la modelización que se presenta en este enunciado, plantear la aplicación del método de *clasificación jerárquica* (*establecer-y-refinar)* para el problema de diagnóstico de enfermedades (no necesariamente del hígado). Para ello, haciendo las suposiciones que se consideren necesario, mostrar ejemplos concretos que ilustren los contenidos de las diversas bases de conocimiento a considerar y poner algún ejemplo de razonamiento.

EJERCICIO 4.4. Considérese una instalación para suministro de agua de regadío y consumo humano formada por componentes tales como: depósitos para almacenamiento de agua, conductos (que a su vez pueden ser canales de transporte al aire libre o tuberías), compuertas y bombas accionadas con energía eléctrica. Se desea diseñar un sistema informático de ayuda al diagnóstico de averías en dicha instalación. Para ello se contempla la posibilidad de utilizar el método de diagnóstico basado en modelo de componentes.

SE PIDE:

Contestar si las siguientes frases corresponden a alguno o algunos de los tipos de conocimiento que maneja dicho método indicando de cuáles se trata.

S1. “La instalación tiene los depósitos D1, D4 y D5 además de otros 7 depósitos. Estos tres depósitos son de gran capacidad por lo que presentan en ocasiones (normalmente una vez al año) problemas de contaminación biológica por crecimiento de algas, lo que no ocurre normalmente en el resto de los depósitos.”

S2. “Las bombas B4, B5 y B6 son de la firma Energizer. Cuando una bomba de este tipo entra en funcionamiento produce un caudal a su salida de 25 metros cúbicos por segundo. Si opera correctamente, el caudal a la entrada de la bomba debe ser el mismo.”

S3. “Las averías correspondientes a las bombas no pueden ser reparadas por el equipo propio de mantenimiento por lo que deben encargarse a equipos especializados lo que supone un incremento importante de coste.”

S4. “Los depósitos D2 y D4 están ubicados en el centro de la ciudad al igual que el equipo de mantenimiento. Los depósitos D7 y D9 están en poblaciones lejanas (a más de 50 Km).”

S5. “Si se observan partículas en suspensión en el agua a la salida del conducto principal entonces puede existir avería en la bomba B17 o contaminación por algas en el depósito D4.”

S6. “Si el conducto C9 se encuentra en buen estado (es decir, no presenta fugas de agua ni roturas) el caudal a la entrada es el mismo que a la salida. Cuando dicho conducto tiene problemas de juntas gastadas, entonces presenta fugas lo que hace que a la salida se reduzca el caudal en 0.5 m3/seg”.

Explicar si es posible representar las siguientes sentencias con el método de diagnóstico basado en modelo de componentes explicando con algún ejemplo aproximado cómo podría realizarse:

S7. "Cuando se abre la compuerta C9 se vacía el depósito D3".

S8. "La tubería T3 sirve para mantener con el mismo nivel los depósitos D3 y D5".

Considerar ahora el método de diagnóstico cubrir y diferenciar. Responder cuáles de las sentencias anteriores (de la S1 a la S8) pueden considerarse como alguno o algunos de los tipos de conocimiento que maneja el método de cubrir y diferenciar, indicando de qué conocimiento se trata en cada caso.

EJERCICIO 4.5. Considerar un caso de un sistema de diagnóstico basado en modelo de componentes en donde se tiene un conjunto de variables X1, X2, …, X9 con los posibles valores {a,b} y los componentes A, B, C, D, E, F, G, H. Cuando se aplica el método, tras el primer paso de inferencia del proceso de razonamiento se tiene el siguiente estado ATMS:

```
<X1=a,{{}}>
<X2=b,{{}}>
<X3=a,{{}}>
<X4=b,{{}}>
<X5=a,{{}}>
<X6=a,{{A}}>
<X6=b,{{B,C}}>
<X7=a,{{C}, {A}}>
<X7=b,{{D,C}, {E, F}}>
<X8=a,{{A}}>
<X8=b,{{B, E}, {A, C}}>
<X9=a,{{A, C}}>
<X9=b,{{D}}>
<no-good, {{A, B, C, E}, {A, C, F}, {B, E, G}}>
```

SE PIDE:

1. ¿Cuáles son las medidas observadas? ¿Por qué?

2. ¿Cuáles son los conflictos? ¿Por qué?

3. ¿Cuáles son los candidatos? ¿Por qué?

4. ¿Qué medidas deben observarse si están averiados los componentes C y E y los demás operan correctamente?

5. ¿Cuál es la siguiente observación a realizar? ¿Por qué?

5 Diseño paramétrico

Diseño paramétrico es un caso particular de problema de configuración en donde el sistema a diseñar se considera con una estructura constante. En este capítulo se presenta el método denominado *proponer y revisar* como una forma de realizar diseño paramétrico. Este método emula el proceso de tanteo que realizan las personas para proponer configuraciones que son revisadas en sucesivos pasos aplicando soluciones heurísticas hasta garantizar la coherencia global de la configuración final.

5.1 Descripción general

En general, los problemas de configuración se aplican a sistemas de componentes, es decir, sistemas que se describen con un conjunto de componentes que forman una estructura. Para expresar las características que debe cumplir el sistema a configurar

se definen (1) *requisitos funcionales,* es decir, indicaciones sobre las funciones que debe desempeñar el sistema a diseñar y (2) *preferencias* que incluyen características no funcionales que debe tener el sistema (por ejemplo, limitación del coste de construcción o restricciones de instalación). El *diseño* se expresa como un conjunto de componentes detallados (es decir, cada componente se describe por su tipo y los valores de un conjunto de atributos que lo caracterizan) y organizados en una determinada estructura. El problema de configuración se define la siguiente forma:

> **Definición 5.1:** *Configurar* un sistema de componentes tiene como objetivo encontrar un diseño (una estructura de componentes detallados) que satisface unos determinados requisitos funcionales y preferencias.

Algunos problemas pueden contemplarse también como problemas de configuración, por ejemplo:

- *Especificar* un sistema de componentes en donde los datos de entrada son intenciones que expresan los principios generales en los que se basa la definición de unos requisitos funcionales detallados. Las intenciones representan una clase general de requisitos pero definidos de forma imprecisa que debe detallarse mediante interacción con el usuario y mediante cierto conocimiento sobre las posibles características técnicas del sistema a diseñar

- *Planificar* una actuación entendida como un sistema de componentes. A partir de un conjunto de requisitos funcionales y preferencias sobre la actuación se genera como resultado del diseño de un plan (ver el siguiente capítulo para una discusión más detallada sobre la relación entre planificación y configuración).

El método que se describe en este capítulo resuelve un subtipo de problema de configuración en donde se asume que el sistema a diseñar tienen una estructura constante. Dicha estructura puede definirse mediante un conjunto prefijado de parámetros que caracterizan sus componentes. El proceso de configuración en este contexto recibe el nombre de diseño paramétrico dado que se basa en realizar una búsqueda de los valores de dichos parámetros de forma que se satisfagan ciertas condiciones de partida. Cada diseño concreto diferirá de otro en los valores obtenidos para dichos parámetros, pero todos los diseños tendrán los mismos parámetros descriptivos.

Este método fue propuesto en el sistema SALT (SALT: A K*n*owledge *Ac*quisiton *L*anguage for Propose-and-Revise Systems, *NaCl*) por Sandra Marcus del equipo de John McDermott en 1985 [Marcus, McDermott, 89]. Chandrasekaran plantea un enfoque más general denominado *proponer-criticar-modificar* [Chandrasekaran, 90] en el que, además de otros métodos, puede incluirse el método aquí descrito. Sobre el método proponer y revisar pueden también consultarse los trabajos de Motta y Zdrahal sobre aspectos de eficiencia del método [Zdrahal, Motta, 95; Motta, Zdrahal, 96].

Como ejemplo ilustrativo del este método puede considerarse el problema de diseño de la mecánica de un ascensor que fue abordado en la construcción del sistema VT (*Vertical Transport*) [Marcus et al., 87] (figura 5.1). De dicho problema se encuentra amplia documentación dado que, aunque se desarrolló inicialmente con el sistema SALT, sirvió de base dentro del proyecto Sisyphus II orientado a unificar diversas metodologías de ingeniería del conocimiento [IJHCS, 96]. En este problema la estructura de la mecánica del elevador es constante ya que se manejan siempre los mismos tipos de componentes (un motor, una cabina, un cable de carga, un cable de gobierno, un contrapeso, etc.). El problema de diseño está en encontrar los valores concretos que caracterizan dichos componentes para unas determinadas

especificaciones (modelo de cabina, dimensiones de la cabina, potencia del motor, tipo de cable, etc.).

El método utiliza la siguiente terminología:

- *Parámetros*: cada parámetro representa una característica significativa (estructural, funcional, etc.) del sistema a diseñar. Los parámetros pueden tomar valores numéricos o cualitativos. Por ejemplo, en el caso de VT se manejan parámetros como los siguientes:

```
capacidad(cabina)              [2000 ... 4000] libras
velocidad(cabina)              {200,250,300,350,400} pies/min
modelo(puerta)                 {ssso, 2sso, ssco, 2sco}
apertura(puerta)               {central, lateral}
ancho(plataforma)              N pulgadas
largo(plataforma)              N pulgadas
modelo(plataforma)             {2.5B, 4B, 6B}
longitud(cable-gobierno)       N pulgadas
diámetro(cable-gobierno)       R pulgadas
```

- *Especificaciones*: son parámetros que corresponden a los datos de partida de la búsqueda de la solución. Incluyen requisitos funcionales, es decir, indicaciones sobre las funciones que debe desempeñar el sistema a diseñar, y preferencias sobre características no funcionales que debe tener el sistema (por ejemplo, limitación del coste de construcción o instalación). En el caso de VT se manejan un total de 23 parámetros de especificación. Por ejemplo, para un determinado caso de diseño se pueden tener parámetros para describir las especificaciones como los siguientes:

```
velocidad(cabina) = 250 pies/minuto
plantas(edificio) = 8 plantas
apertura(puerta)  = central
ancho(plataforma) = 70 pulgadas
ancho(plataforma) = 84 pulgadas
```

- *Parámetros finales*: son parámetros que recogen las características estructurales del resultado del proceso de configuración. Corresponden a un conjunto prefijado de parámetros con los que se expresa la solución del problema. En VT se tienen un total de 35 parámetros finales como los siguientes

```
modelo(amortiguador-contrapeso) =    OH-1
ancho(marco-contrapeso) =            31 pulgadas
diámetro(cable-gobierno) =           0.375 pulgadas
modelo(cable-carga) =                (3)-0.5
peso(suplemento-carga-cabina) =      500 libras
```

- *Parámetros intermedios*: son parámetros utilizados durante el proceso de razonamiento para poder deducir los valores de los parámetros finales a partir de las especificaciones. Dichos parámetros pueden ser utilizados para mostrar explicaciones de cómo se ha llegado a la solución propuesta. En VT se manejan un total de 168 parámetros intermedios.

Se considera que el conocimiento que relaciona dichos parámetros, debido a su complejidad, establece relaciones que se pueden entender de dos tipos: (1) relaciones en sentido directo que van de las especificaciones a los parámetros finales a través de los intermedios, y (2) relaciones en sentido inverso que van en la dirección opuesta. Para obtener los valores de los parámetros finales, el proceso se ordena en dos pasos fundamentales:

1. *Proponer*. El proceso de configuración parte de las especificaciones y obtiene los valores de otros parámetros haciendo uso de conocimiento que los relaciona en sentido directo (por ejemplo, fórmulas aritméticas o relaciones condicionales en forma de reglas). Este proceso se repite encadenándose por el conjunto de parámetros hasta llegar a los parámetros finales.

2. *Revisar*. En un segundo paso, utilizando conocimiento expresado a otro nivel, se revisan las relaciones en sentido inverso con el fin de comprobar que se satisfacen. Si no es así, entonces se lleva a cabo una modificación del valor de uno o más parámetros y se vuelve a realizar el proceso.

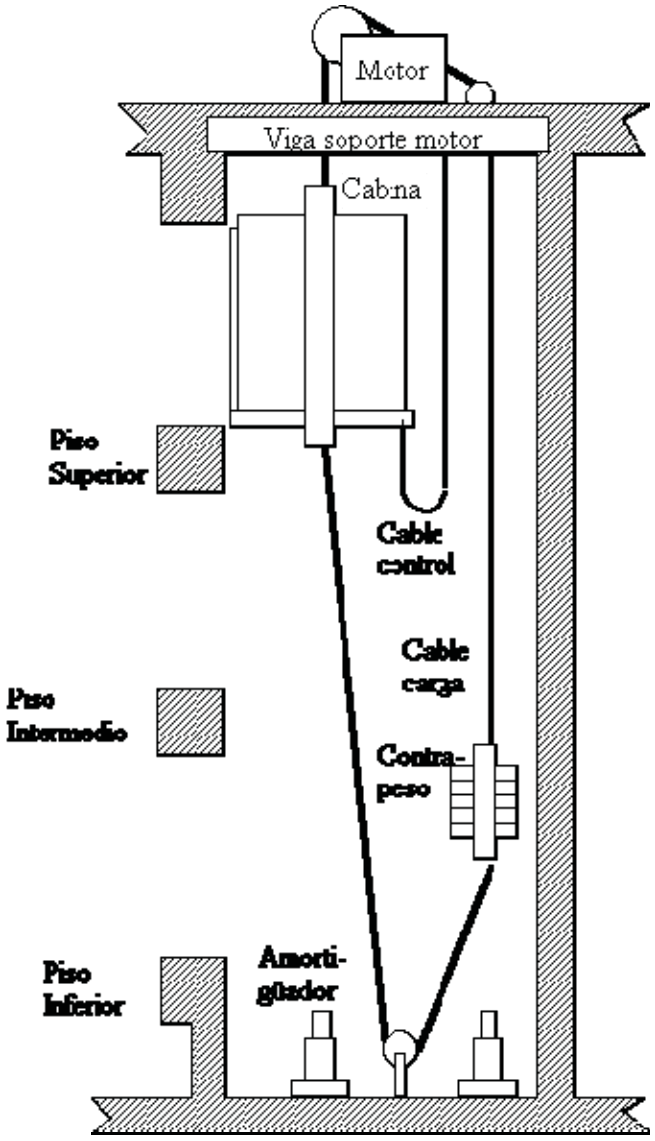
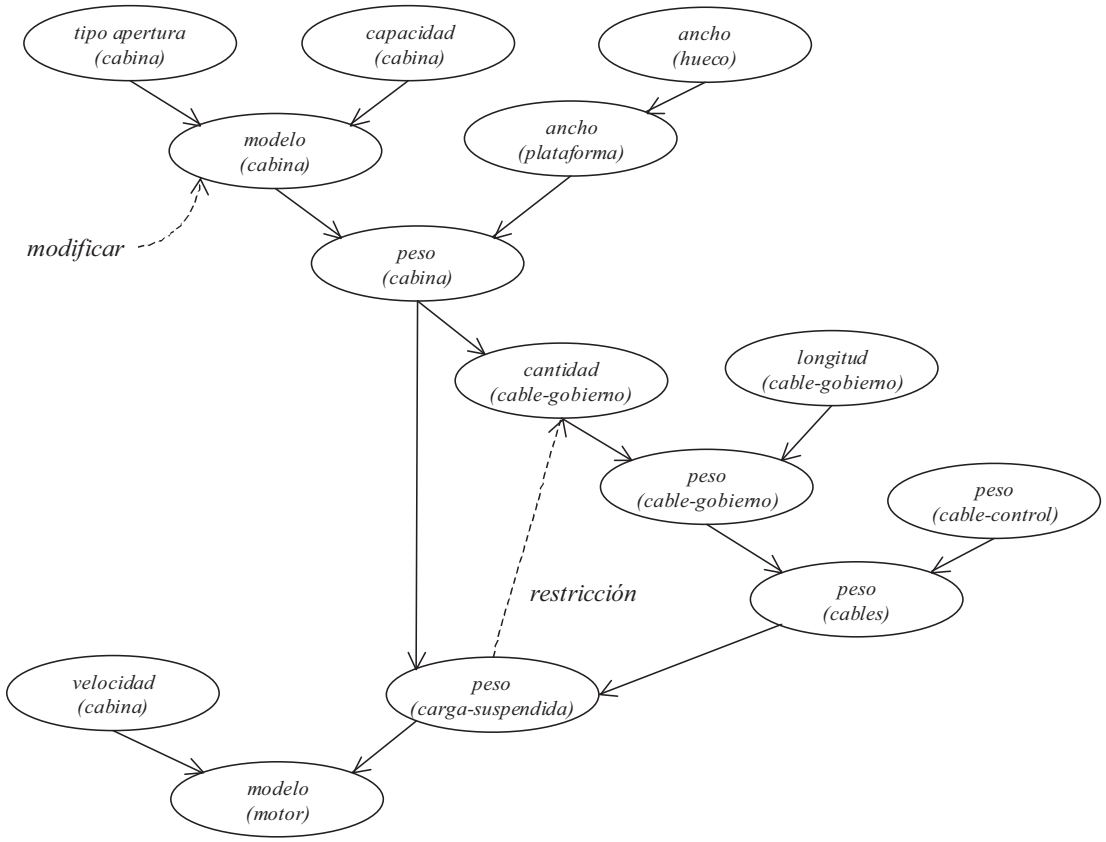


Figura 5.1: Ejemplo de problema de configuración: diseño de la mecánica de un ascensor [Yost, 92]



5.2: Ejemplo de la forma de relacionar y actuar sobre parámetros en proponer y revisar.

La forma de actuación de este método tiene cierta similitud con la que tradicionalmente se denomina generar-y-probar (*generate and test*). Pero, como diferencia esencial, el procedimiento correspondiente a generar-y-probar separa el conocimiento en dos áreas independientes de forma que, tras el paso de generación de una hipótesis, se realiza la prueba de la hipótesis y, si no resulta satisfactoria, se genera otra hipótesis sin que la prueba realizada a la anterior hipótesis influya en dicha generación [Fensel, 00]. En el caso de proponer-y-revisar, por el contrario, el paso de proponer genera una hipótesis de diseño y, después, el resultado de la revisión influye de forma directa en la forma que se va a realizar la siguiente propuesta de diseño, produciéndose una realimentación (que no existe en generar-y-probar) y que constituye la base del proceso de búsqueda de la solución.

5.2 Organización del conocimiento

El método proponer y revisar maneja cuatro tipos de conocimiento: cálculo de parámetros, restricciones de diseño, modificaciones y preferencias de cambios. El conocimiento sobre *cálculo de parámetros* indica cómo determinar los valores iniciales de unos parámetros a partir de otros. Estos valores se consideran iniciales dado que, durante el proceso de diseño, pueden ser modificados con el fin de satisfacer criterios adicionales. Este conocimiento puede visualizarse en forma de red en donde cada nodo es un parámetro y cada arco que concluye sobre un parámetro Y parte de un número de nodos $X_1, ..., X_n$ indicando con ello que el parámetro Y se calcula a partir de los parámetros $X_1, ..., X_n$. La red tiene como requisito que no debe presentar bucles. También se asume que la forma de la red debe ser tal que el valor deducido para cada parámetro debe ser único.

La forma concreta de hacer cada cálculo puede utilizar una formulación matemática con una notación funcional o relaciones para deducir valores cualitativos expresadas con un conjunto de reglas. Por ejemplo, siguiendo con el caso del sistema VT se pueden incluir afirmaciones como la siguiente:

```
ancho(plataforma) = X y largo(plataforma) = Y
→ peso(cabina) = 130*(X + Y)/12
```

Dicha regla indica cómo calcular el peso de la cabina mediante una fórmula aritmética sencilla a partir de otros dos parámetros numéricos. También se pueden tener reglas para obtener el valor cualitativo de un parámetro a partir de los valores (numéricos o cualitativos) de otros parámetros. Por ejemplo:

```
ancho(plataforma) < 93 →  modelo-seguridad(cabina)=B1
ancho(plataforma) = [93,114] →  modelo-seguridad(cabina)=B4
ancho(plataforma) > 114 →  modelo-seguridad(cabina)=B6
```

El supuesto de que la red debe permitir obtener un único valor deducido como máximo para cada parámetro implica que si existe una regla tal como A=a, B=a → C=b, entonces no debería existir una regla tal como A=a, B=a → C=d, es decir, no deben existir reglas con las mismas premisas pero diferente conclusión para el mismo parámetro.

La base de conocimiento de *restricciones de diseño* recoge las relaciones entre parámetros y condiciones locales que deben verificarse y no están incluidas en el conocimiento de cálculo de parámetros. Se describe en forma de expresiones que pueden relacionan valores de unos parámetros con otros. En dichas expresiones se pueden indicar valores límite (superior o inferior) de parámetros individuales. Por ejemplo:

```
restricción-máxima-altura-cabina: altura(cabina) < 240
```

La restricción se puede representar con un nombre (`restricción-máxima-altura-cabina`) y con un cuerpo en donde se establece la relación que debe cumplirse. Se dan también restricciones que incluyen varios parámetros y que, opcionalmente, pueden estar condicionadas para que se consideren sólo en caso de que se satisfaga cierta expresión lógica. Por ejemplo:

```
modelo(polea-motor) = K3140
→
restricción-relación-ángulo-tracción-motor:
tracción(motor) < 0.007888 * ángulo(motor) + 0.675
```

En dicha expresión se indica que si el modelo de polea de motor es K3140 entonces debe cumplirse una restricción que establece una determinada relación en forma de desigualdad entre el índice de tracción del motor y el ángulo de contacto del motor.

La base de conocimiento de *modificaciones* representa criterios heurísticos que indican los cambios que pueden realizarse en los parámetros para solucionar violaciones de restricciones que se hayan encontrado. Si se representa en reglas, el antecedente puede tener el nombre de la restricción que se incumple y el consecuente la acción a realizar para intentar resolver la violación. Por ejemplo:

```
VIOLACIÓN(restricción-máxima-altura-pila-contrapeso))
→ SIGUIENTE(distancia(raíles))

VIOLACIÓN(restricción-máximo-índice-tracción-motor)
→ DECREMENTAR(distancia-contrapeso(plataforma), 0.5)

VIOLACIÓN(restricción-máximo-índice-tracción-motor)
→ INCREMENTAR(suplemento-carga(cabina), 100)

VIOLACIÓN(restricción-máximo-índice-tracción-motor)
→ SIGUIENTE-CRITERIO(modelo(compensación-cable),calidad)
```

En dichos ejemplos, las modificaciones indican cómo sustituir el valor en curso de un determinado parámetro. Por ejemplo, si el parámetro es numérico se puede indicar mediante acciones `INCREMENTAR(X,K)` o `DECREMENTAR(X,K)` para indicar que el parámetro *X* incremente o decremente respectivamente el valor actual en un determinado número *K*. Si el parámetro es cualitativo se puede indicar `SIGUIENTE(X)` o `ANTERIOR(X)` para indicar que se elija el siguiente o el anterior valor del parámetro *X* dentro de un determinado orden prefijado de valores. También puede utilizarse la representación `SIGUIENTE-CRITERIO(X,C)` o `ANTERIOR-CRITERIO(X,C)` para elegir el siguiente o el anterior valor del parámetro *X* atendiendo a un cierto criterio específico que permita manejar diferentes ordenaciones de los valores posibles (por ejemplo *C* = calidad, seguridad, coste, tamaño, etc.).

En el sistema original VT se manejaban *acciones iterativas*. Estas acciones corresponden a acciones de incremento o decremento de un parámetro y se aplican de forma iterativa repitiéndose hasta que se resuelve la violación o hasta que se llega a una determinada condición. Para ello es conveniente considerar también nombres de acciones como la siguiente:

```
VIOLACIÓN(restricción-máxima-altura-pila-contrapeso)
→ INCREMENTAR-HASTA(altura(marco),6,
                    restricción-máxima-altura-marco)
```

en donde la acción `INCREMENTAR-HASTA(`*X*`, `*K*`, `*R*`)` incrementa el parámetro *X* en la cantidad *K* repitiendo el proceso mientras que la violación que ha provocado esta acción se mantenga y comprobando que además no se viola la restricción *R*. No obstante, en los ejemplos del presente texto, se manejarán únicamente acciones que se aplican de una sola vez.

1.	*No causa problemas*
2.	*Incrementa requisitos de mantenimiento*
3.	*Hace la instalación más difícil*
4.	*Pequeño cambio del tamaño del equipo*
5.	*Pequeña violación sobre equipo*
6.	*Pequeño cambio de especificaciones*
7.	*Requiere un especial diseño*
8.	*Grandes cambios en tamaño de equipo*
9.	*Grandes cambios en el edificio*
10.	*Grandes cambios en especificaciones*

Figura 5.3: Orden de prioridad utilizado en el ejemplo del ascensor.

Tipo de conocimiento		**Explicación**
Cálculo de parámetros	*Significado*	Indica cómo determinar los valores inciales de unos parámetros a partir de otros, utilizando formulación matemática cuantitativa o estructuras lógicas para deducir valores cualitativos
	Representación	Funciones, Reglas
	Ejemplos	`ancho(plataforma) < 93 → modelo-seguridad(cabina)=B1` `ancho(plataforma) > 114 → modelo-seguridad(cabina)=B6`
Restricciones de diseño	*Significado*	Relaciones entre parámetros y condiciones locales que deben verificarse y no están incluidas en el conocimiento de cálculo de parámetros
	Representación	Restricciones, restricciones condicionadas

	Ejemplos	`modelo(polea-motor) = K3140`
		`→ relación-ángulo-tracción-motor:`
		`tracción(motor) < 0.007888 * ángulo(motor) + 0.675`

Modificaciones	*Significado*	Criterios heurísticos que indican los cambios que deben realizarse en los parámetros para solucionar violaciones de restricciones
	Representación	Reglas

	Ejemplos	`VIOLACIÓN(máximo-índice-tracción-motor)`
		`→ DECREMENTAR(distancia-contrapeso(plataforma), 0.5)`
		`VIOLACIÓN(máximo-índice-tracción-motor)`
		`→ INCREMENTAR(suplemento-carga(cabina), 100)`

Preferencias de cambios	*Significado*	Conocimiento de control para selección de modificaciones
	Representación	Tablas de prioridad
	Ejemplos	`PRIORIDAD(ANTERIOR(modelo(polea-motor),tamaño),1)` `PRIORIDAD(INCREMENTAR(peso(suplemento-cabina),X),4)`

Figura 5.4. Organización del conocimiento.

En general, dada una determinada violación se puede tener una, ninguna o varias modificaciones posibles. Por ello, para cuando se dan varias alternativas posibles es importante contar con criterios adicionales que indiquen qué modificación elegir. Para ello se maneja conocimiento de *preferencias de cambios*. Se trata de conocimiento de control que se puede expresar en forma de orden de prioridad. Así, cada cambio posible tendrá asociado un orden de prioridad, que indicará cuándo es preferible frente a otro. En cada dominio particular será posible expresar un orden de prioridad atendiendo a la facilidad con la que se pueden realizar los cambios. Por ejemplo, la figura 5.3 muestra el orden de prioridad considerado en el dominio de mecánica del ascensor. En este caso se manejan 10 niveles de forma que los niveles inferiores se prefieren antes que los superiores. A cada tipo de modificación se le asociará un orden de prioridad de acuerdo con dicha escala. Por ejemplo:

```
PRIORIDAD(ANTERIOR(modelo(cabina)),2)
PRIORIDAD(INCREMENTAR(peso(suplemento-carga-cabina),K),4)
PRIORIDAD(DECREMENTAR(distancia-contrapeso(cabina),K),3)
PRIORIDAD(ANTERIOR-CRITERIO(modelo(polea-motor),tamaño),2)
```

Nótese que el conocimiento de cálculo de parámetros y de restricciones de diseño tiene naturaleza de tipo teórico y, normalmente, está debidamente documentado en los manuales profesionales de diseño. Sin embargo el conocimiento sobre modificaciones y de preferencias de cambios se basa más en criterios derivados de la experiencia en la realización de diseños previos por lo que es de naturaleza heurística. Se trata además de conocimiento de control que dirige el proceso de búsqueda de la solución indicando los caminos más prometedores en cuanto a cambios a realizar para resolver violaciones. Una mejor calidad de dicho conocimiento permitirá alcanzar de una forma más eficiente la solución.

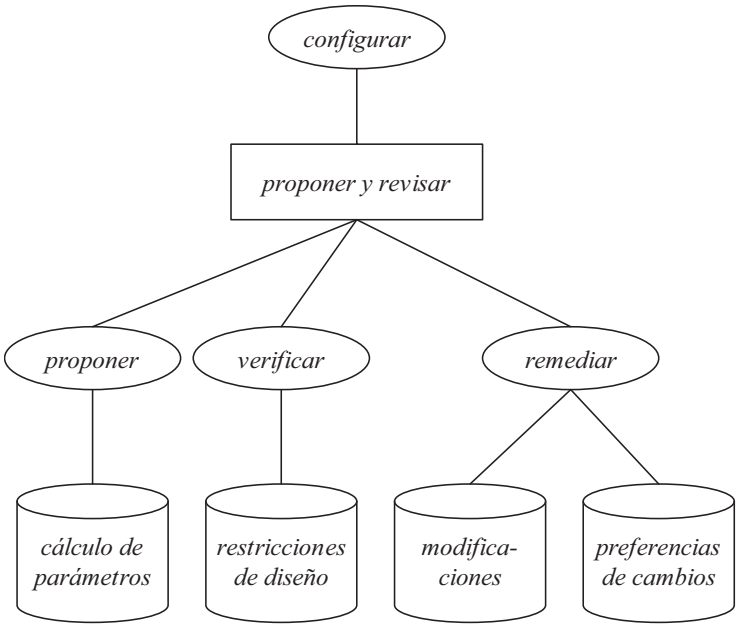


Figura 5.5: Estructura del método considerada en el algoritmo 1.

5.3 Inferencia

En este apartado se describen dos algoritmos. El primero corresponde al algoritmo del método proponer y revisar propiamente dicho. El segundo es una adaptación de dicho método para problemas de asignación.

Rol dinámico	Significado	Representación	Ejemplo
`diseño`	diseño en curso, inicialmente contiene sólo las especificaciones	conjunto de pares parámetro-valor	`{apertura=central, personas=6, plantas=12}`
`cambio`	modificaciones de valores en ciertos parámetros	conjunto de pares parámetro-valor	`{personas=5, apertura=lateral}`
`violaciones`	restricciones de diseño que no se cumplen	conjunto de identificadores de restricciones de diseño	`{R23, R15, R47}`
`cambios`	conjunto de modificaciones, ordenado según preferencias	conjunto de subconjuntos de parámetro-valor (disyunción de conjunciones)	`{{personas=5},{apertura=lateral}` `{personas=5, apertura=lateral}}`
`éxito`	indica si ha tenido éxito la búsqueda del diseño	valor de `{VERDADERO, FALSO}`	`VERDADERO`

Figura 5.6: Roles dinámicos que intervienen en el proceso de inferencia.

5.3.1. Algoritmo 1: Proponer y revisar

Los pasos de inferencia considerados en este método son: proponer, verificar y remediar (figuras 5.5 y 5.7). El paso de inferencia *proponer* se apoya en el conocimiento de cálculo de parámetros y tiene como objetivo obtener valores de los parámetros finales a partir de (1) los valores de los parámetros iniciales, o bien, (2) a partir de modificaciones a un diseño anterior. Recibe como entrada el diseño, es decir, un conjunto de parámetros con valores conocidos, y genera como resultado un nuevo diseño como un conjunto de valores para todos los parámetros. Como

entrada opcional se tiene lo que se denomina *cambio*, es decir, un subconjunto de parámetros con valores que indica el conjunto de modificaciones que se deben realizar al diseño que se da como entrada. En este caso, el paso de inferencia realiza los cambios en los parámetros indicados y, además, propaga cambios a otros parámetros relacionados. El cambio puede afectar tanto a parámetros de especificación como a parámetros intermedios. En el caso de que se modifique el valor de un parámetro intermedio, el paso proponer propaga ese cambio hacia delante, es decir, actualizando sólo los valores que dependen de dicho parámetro según la red existente en la base de cálculo de parámetros.

Por ejemplo, el paso de inferencia puede recibir como dato el diseño {X1=a, X2=b} y el cambio {} (vacío) y genera como resultado el nuevo diseño {X1=a, X2=b, X3=c, X4=b, X5=b, X6=a, X7=b, X8=c, X9=a}. Otra posibilidad es que reciba como entrada el diseño {X1=a, X2=b, X3=c, X4=b, X5=b, X6=a, X7=b, X8=c, X9=a} y el cambio {X2=c, X3=a} generando como resultado el diseño {X1=a, X2=c, X3=a, X4=b, X5=c, X6=a, X7=a, X8=c, X9=a} que se obtiene propagando el cambio a ciertos parámetros según la red de cálculos y manteniendo los valores para el resto de parámetros.

```
INFERENCIA proponer
        DATOS:                  diseño, cambio
        BASES DE CONOCIMIENTO: cálculo-de-parámetros
        RESULTADOS:             diseño

INFERENCIA verificar
        DATOS:                  diseño
        BASES DE CONOCIMIENTO: restricciones-de-diseño
        RESULTADOS:             violaciones

INFERENCIA remediar
        DATOS:                  violación, diseño
        BASES DE CONOCIMIENTO: modificaciones, preferencias-de-cambios
        RESULTADOS:             cambios
```

Figura 5.7: Pasos de inferencia considerados en el algoritmo 1.

Una forma de hacer este proceso a nivel simbólico es manejando un conjunto de reglas que se procesan mediante encadenamiento hacia delante junto a un sistema de mantenimiento de la verdad TMS (*Truth Maintenance System*) del tipo JTMS (*Justification-based TMS*) [Doyle, 79; Forbus, de Kleer, 93]. El TMS almacena cómo unos parámetros obtienen su valor a partir de otros manteniendo una red de dependencias entre ellos denominadas justificaciones. Dado un determinado cambio a realizar en el diseño, por ejemplo cambiar los valores {X2=b, X3=c} por los valores {X2=c, X3=a}, se consulta primero al TMS para que indique cuáles son los hechos que se han derivado de {X2=b, X3=c}. El TMS contesta que {X2=b, X3=c, X5=b, X7=b}. A continuación se borran dichos hechos del diseño en curso y se lanza de nuevo la inferencia en reglas por encadenamiento hacia delante con los cambios {X2=c, X3=a} junto a los hechos que no han sido borrados, es decir con {X1=a, X2=c, X3=a, X4=b, X6=a, X8=c, X9=a} dando como resultado el nuevo diseño {X1=a, X2=c, X3=a, X4=b, X5=c, X6=a, X7=a, X8=c, X9=a}.

El paso de inferencia *verificar* utiliza el conocimiento de restricciones de diseño para comprobar si el diseño satisface dichas restricciones. Recibe como entrada el diseño en curso y genera las violaciones, es decir, las restricciones que no se satisfacen. Por ejemplo puede recibir como entrada el diseño {X1=a, X2=c, X3=a, X4=b, X5=c, X6=a, X7=a, X8=c, X9=a} y genera el conjunto de violaciones {R3, R5}.

El paso de inferencia *remediar* tiene como fin determinar las modificaciones posibles que pueden resolver una determinada violación. Recibe como entrada la violación y el diseño en curso y, utilizando el conocimiento de modificaciones y de preferencias de cambios, genera un conjunto ordenado de opciones posibles a considerar para eliminar la violación.

Cuando se trata de solucionar una determinada restricción, las modificaciones se escogen de la siguiente forma. Primero se eligen de una en una, escogiendo en

primer lugar las que tengan más prioridad (ante igualdad de niveles de prioridad se toman al azar). Si ninguna de estas soluciones de forma individual permite llegar a una solución, entonces se eligen grupos de modificaciones eligiendo primero los conjuntos con menor número de elementos y, entre conjuntos del mismo tamaño, aquel cuyo nivel de prioridad global sea más preferente. El nivel de prioridad global puede entenderse, por ejemplo, como la suma de los niveles de prioridad de los elementos, o el máximo de todos ellos, etc.

```
METODO proponer-y-revisar
  DATOS: especificaciones
  RESULTADOS: diseño, éxito

ALGORITMO
1.   diseño := especificaciones
2.   cambio := ϕ
3.   generar-propuesta(diseño, cambio -> diseño, éxito)

PROCEDIMIENTO generar-propuesta
  DATOS: diseño, cambio
  RESULTADOS: diseño, éxito
1.   proponer(diseño, cambio -> diseño)
2.   verificar(diseño -> violaciones)
3.   IF violaciones = ϕ
4.   THEN éxito := VERDADERO
5.   ELSE
6.     GET(violación, violaciones)
7.     resolver-violación(diseño, violación -> diseño, éxito)

PROCEDIMIENTO resolver-violación
  DATOS: diseño, violación
  RESULTADOS: diseño, éxito
1.   remediar(violación, diseño -> cambios)
2.   IF cambios = ϕ
3.   THEN éxito := FALSO
4.   ELSE
5.     REPEAT
6.       GET(cambio, cambios)
7.       generar-propuesta(diseño, cambio -> diseño, éxito)
8.     UNTIL (éxito = VERDADERO) OR (cambios = ϕ)
```

Figura 5.8: Ejemplo de algoritmo del método de proponer y revisar.

Por ejemplo, supóngase que se recibe como datos de entrada la violación con el identificador de restricción R3 junto al diseño en curso {X1=a, X2=c, X3=a, X4=b, X5=c, X6=a, X7=a, X8=c, X9=a} y se tienen las siguientes reglas de modificación:

```
M5: VIOLACIÓN(R3) → SIGUIENTE(X3)
M6: VIOLACIÓN(R3) → ANTERIOR(X2)
```

Y además las siguientes preferencias:

```
P7: PRIORIDAD(SIGUIENTE(X3),2)
P9: PRIORIDAD(ANTERIOR(X2),4)
```

Esto genera como resultado el conjunto de cambios {{X3=b},{X2=b},{X2=b, X3=b}} que indica que, en primer lugar debe intentarse el cambio X3=b (siguiente valor de X3=a), en segundo lugar X2=b (anterior valor de X2=c) y, en caso de que fallen ambos, debe intentarse en último lugar la combinación { X2=b, X3=b}.

La figura 5.8 muestra en pseudo-código el ejemplo de algoritmo considerado, cuya estructura de inferencia se ilustra en la figura 5.9. En este algoritmo los puntos de fallo para realización de vuelta atrás se dan cuando no se encuentran cambios para resolver una restricción.

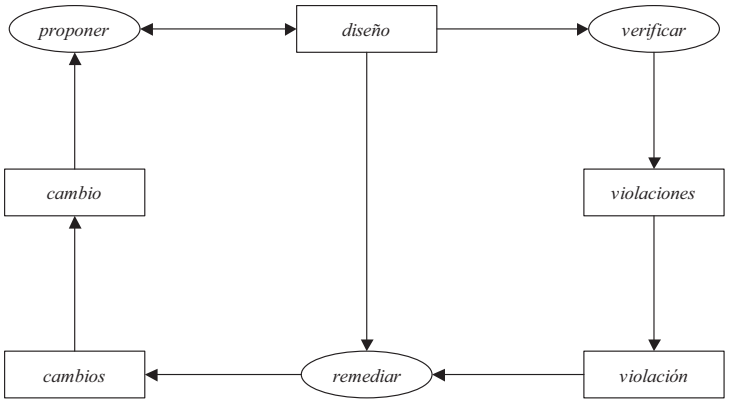


Figura 5.9: Estructura de inferencia del ejemplo de algoritmo 1.

Aspectos de búsqueda y eficiencia del método

El algoritmo aquí presentado en este capítulo sigue la estrategia denominada en la literatura sobre el método proponer y revisar como CMR (*Complete Model then Revise*) que propone el diseño para todos los parámetros antes de realizar la revisión. La estrategia que utilizaba el sistema VT originalmente con SALT se denomina EMR (*Extend Design then Revise*) en donde cada vez que se toma una decisión parcial de diseño se revisan las restricciones. La elección de una u otra estrategia dependerá de la forma en que se realiza de forma natural por los profesionales expertos y de la posibilidad de analizar información determinante para dirigir la búsqueda (por ejemplo, la estrategia CMR es más adecuada tal como se plantea en [Zdrahal, Motta, 95] para tratar de forma conjunta violaciones).

El algoritmo de proponer y revisar aquí mostrado utiliza conocimiento del dominio para restringir la búsqueda tal como se explica a continuación. El espacio total de diseños posibles viene dado por los parámetros y por los valores posibles que pueden tomar dichos parámetros. Cuando un diseño no satisface una restricción, las posibilidades de solución de la violación vienen dadas por los cambios en los valores de los parámetros de la restricción y, también, en los parámetros a partir de los cuales se calculan los parámetros de la restricción. Sin embargo, en vez de considerar como cambios posibles todos estos parámetros, se contemplan sólo los casos expresados por las modificaciones, lo cual supone una reducción importante de combinatoria. Normalmente dichos casos de modificación son los que se corresponden con los cambios aceptables en el diseño y que reflejan los grados de libertad reales que tiene el diseñador. Por ejemplo, si para resolver la violación de una determinada restricción se tuvieran como candidatos el número de plantas del edificio o la potencia del motor, evidentemente, sólo se contemplará como posible modificación la potencia del motor y no el número de plantas del edificio que no puede variarse.

La versión de algoritmo aquí presentada realiza una búsqueda en profundidad con *bactracking* cronológico. Así, cuando se produce un bloqueo en un determinado punto, se hace vuelta atrás (*backtracking*) a las alternativas de modificación más recientes, es decir, de mayor profundidad y, después, se elige entre ellas de acuerdo con las medidas de prioridad. Este proceso de exploración como ya es conocido permite encontrar una solución aunque no se garantiza que sea óptima. En este contexto, una solución óptima se entiende como una solución que se alcanza mediante un camino formado por un conjunto de modificaciones en donde todas ellos no superan una medida *M* mínima de orden prioridad y no existe otra solución con un camino con medida *M'* menor que *M.*

Otro aspecto interesante de la búsqueda es la forma en que se intentan resolver las restricciones. Se toman de una en una y no se consideran puntos de vuelta atrás. Con esta simplificación se está asumiendo que si la violación de una restricción *R1* no se puede resolver con sus propias modificaciones, entonces no será posible resolverla mediante las modificaciones de otra restricción *R2*. Dicho supuesto se cumple habitualmente y supone una poda importante de opciones de búsqueda que habitualmente producen caminos de fallo, lo que implica una mejora significativa de eficiencia. Si este supuesto no se diera en algún caso, se puede conseguir haciendo que, en la base de conocimiento de modificaciones, las modificaciones de *R1* incluyan las modificaciones de *R2* que resuelven la violación de *R1*.

Para un caso general, el algoritmo aquí presentado debe ser extendido al menos para producir fallo también cuando se detectan bucles, es decir, cuando por efecto de un cambio se vuelve a un diseño considerado ya en una etapa anterior. Para ello, en una resolución manual o cuando la dimensión del problema no es importante se puede llevar un registro de los diseños que se han realizado hasta el momento de forma que cuando se verifique un diseño se compruebe que no esté ya presente en el alguna etapa previa del camino de búsqueda. Este proceso obliga a revisar cada estado nuevo con respecto al total de estados que forman el camino, lo cual puede

ser costoso en problemas con cierta dimensión en donde los estados pueden tener muchos parámetros y la profundidad del camino pueda crecer significativamente (aunque normalmente dicha profundidad debe acotarse por un número máximo de niveles). Sandra Marcus plantea tratar este problema en SALT mediante identificación de restricciones antagonistas de acuerdo con la siguiente definición:

> **Definición 5.2:** Una restricción *R1* es *antagonista* con respecto a otra *R2* si para remediar *R1* se aplica un remedio opuesto al remedio que se propone para remediar *R2*.

Ejemplos de remedios opuestos son {`INCREMENTAR(X,K1), DECREMENTAR(X,K2)`} o {`SIGUIENTE(X), ANTERIOR(X)`}. De acuerdo con esta definición, mediante un análisis de la restricciones y de las modificaciones previo a la resolución de problemas es posible detectar conjuntos de restricciones antagonistas. El tratamiento de restricciones de este tipo durante ejecución permite incluir las mejoras como las siguientes:

- Durante la resolución del problema, el sistema almacena las modificaciones realizadas para resolver cada violación de restricción R_i que pueden presentar antagonismo con alguna otra. Si, durante la resolución del problema, se viola otra restricción R_j que es antagonista con R_i, se pueden elegir primero soluciones que no sean opuestas a las modificaciones realizadas para resolver la violación de R_i.

- Tras el paso de verificación, en vez de considerar las restricciones de una en una, se pueden tratar grupos de restricciones antagonistas buscando soluciones que eliminen el máximo grupo de restricciones sin afectar negativamente a otras.

- Cuando no es posible encontrar soluciones que eliminen dos violaciones de restricciones antagonistas, el sistema SALT plantea, al igual que otros enfoques [Descotte, Latombe, 85], manejar conocimiento de control que ordena las restricciones por preferencia. El uso de este conocimiento permite plantear la relajación de ciertas restricciones menos prioritarias de forma que si, al final no es posible encontrar un diseño que cumpla todas las restricciones, al menos se mantendrán las más prioritarias. Además, en ocasiones es posible hacer un tratamiento local de forma que eliminando la restricción más prioritaria se reduzca al mínimo la violación de la restricción antagonista. Por ejemplo se pueden tener dos restricciones de valores máximos `R1:A<10`, y `R2:B<20` en donde *R1* es más prioritaria que *R2*. El tratamiento local debe intentar que, manteniéndose *A* por debajo de 10 (por ejemplo, *A*=9.9) se acerque en lo posible *B* a 20 (por ejemplo *B*=22.3). No obstante, este razonamiento para tratar situaciones de compromiso no fue abordado de forma completa por la herramienta SALT.

El sistema VT original, además de las simplificaciones para mejorar la eficiencia en la búsqueda que ya han sido comentadas, aplica la estrategia de un solo paso de anticipación (*one-step look-ahead*) que descarta una modificación si introduce una nueva violación (esta estrategia no se ha incluido en el algoritmo de proponer y revisar presentado en este capítulo). Dicha estrategia tiene el problema de que poda caminos que potencialmente pueden llevar a la solución como el siguiente: (1) la restricción A es violada, (2) la modificación M elimina la violación de A pero viola la restricción B, (3) la modificación N elimina la violación de B. Por ello, esta simplificación ha sido señalada y criticada por diversos autores [Zdrahal, Motta, 96; Fensel, 95] dado que provoca que, de forma general, la estrategia sea incompleta. Como indica [Fensel, 00] la completitud se puede garantizar sólo si se introducen dos supuestos en el conocimiento del dominio: (1) el conocimiento utilizado por el paso proponer siempre da el mejor diseño (parcial) posible y (2) el

conocimiento de revisar siempre modifica un diseño que viola restricciones con el mejor diseño válido.

El sistema VT comenzó utilizarse en abril de 1985 con 1300 reglas por ingenieros de la empresa *Westinghouse Elevator Company* y llegó a tener en verano de 1986 un total de 2191 reglas del dominio generadas por SALT además de otras 932 reglas generales de operación del método. La mayor parte de los remedios tenían una única forma de remedio o un número reducido, lo que hacía que la búsqueda fuera eficiente. Por ejemplo, de 52 violaciones de restricciones con remedios, 37 tienen un único remedio, 10 tienen dos remedios, 3 tienen tres remedios, 2 tienen cuatro con varias posibilidades de instanciación.

5.3.2. Algoritmo 2: Proponer e intercambiar

El método proponer y revisar puede adaptarse de forma sencilla para resolver problemas de asignación. El problema general de asignación considera dos conjuntos: el conjunto de *recursos* y el conjunto de *necesidades* (por ejemplo, en el caso de aparcamiento de autobuses en una zona turística los recursos son las plazas y las necesidades son los propios autobuses). El problema se define de la siguiente forma:

> **Definición 5.3:** El problema de *asignación* consiste en asociar a cada elemento del conjunto de necesidades un elemento del conjunto de recursos de forma que se satisfagan un conjunto de criterios generales.

La forma de adaptar el algoritmo general de proponer y revisar a este problema es haciendo que el conjunto de parámetros coincida con el conjunto de necesidades y el conjunto de valores posibles que pueden tomar dichos parámetros es el mismo para todos ellos y coincide con el conjunto de recursos. De forma dual, también es

posible plantearlo a la inversa en donde el conjunto de parámetros sea el conjunto de recursos y los valores posibles coincidan con el conjunto de necesidades. En la elección de uno u otro enfoque se tendrá en cuenta que cada parámetro puede tomar como mucho un único valor. Además de estos parámetros puede haber otros adicionales con otros dominios de valores que sean útiles para establecer restricciones que se deben satisfacer. El método así considerado recibe el nombre de *proponer e intercambiar*, dado que lo que se plantea tras una asignación inicial es el intercambio de valores en los parámetros correspondientes a las necesidades.

Rol dinámico	**Significado**	**Representación**	**Ejemplo**
`asignación`	asignación en curso, formada por los parámetros correspondientes a la asignación además de otros secundarios	conjunto de pares parámetro-valor	`{plaza-3=autobús-1, plaza-8=autobús-2, plaza-4=autobús-3, plaza-2=autobús-4, plazas-libres=4}`
`intercambio`	intercambio de los valores de pares de parámetros	conjunto de tuplas de dos parámetros <parámetro-1, parámetro-2>	`{<plaza-3, plaza-4>, <plaza-5, plaza-7>}`
`violaciones`	restricciones de que no se cumplen	conjunto de identificadores de restricciones	`{R23, R15, R47}`
`intercambios`	conjunto de intercambios ordenado según preferencias	conjunto de conjuntos de tuplas de dos parámetros <parámetro-1, parámetro-2> entendido como disyunción de conjunciones	`{{<plaza-3, plaza-4>}, {<plaza-5, plaza-7>}, {<plaza-3, plaza-4>, <plaza-5, plaza-7>}}`
`éxito`	indica si ha tenido éxito la búsqueda	valor de `{VERDADERO, FALSO}`	`VERDADERO`

Figura 5.10: Roles dinámicos que intervienen en el proceso de inferencia del algoritmo 2.

Las figuras desde 5.10 a 5.13 muestran la adaptación la versión correspondiente al método de proponer e intercambiar. Es muy similar al caso de proponer y revisar, con el cambio de que se maneja una asignación en vez de un diseño. Además, el paso de inferencia de proponer se divide en dos: (1) un paso denominado *proponer-inicial* para realizar una propuesta inicial en donde habitualmente se aplica una determinada estrategia de asignación global relativamente sencilla, y (2) el paso *proponer* propiamente dicho que se encarga de la aplicación de los intercambios

correspondientes y que se puede basar en un proceso relativamente sencillo de modificación de la asignación en curso.

```
INFERENCIA proponer-inicial
       DATOS:                -
       BASES DE CONOCIMIENTO: cálculo-de-parámetros
       RESULTADOS:           asignación

INFERENCIA proponer
       DATOS:                asignación, intercambio
       BASES DE CONOCIMIENTO: cálculo-de-parámetros
       RESULTADOS:           asignación

INFERENCIA verificar
       DATOS:                asignación
       BASES DE CONOCIMIENTO: restricciones-de-diseño
       RESULTADOS:           violaciones

INFERENCIA remediar
       DATOS:                violación, asignación
       BASES DE CONOCIMIENTO: modificaciones, preferencias-de-cambios
       RESULTADOS:           intercambios
```

Figura 5.11: Pasos de inferencia considerados en el algoritmo 2.

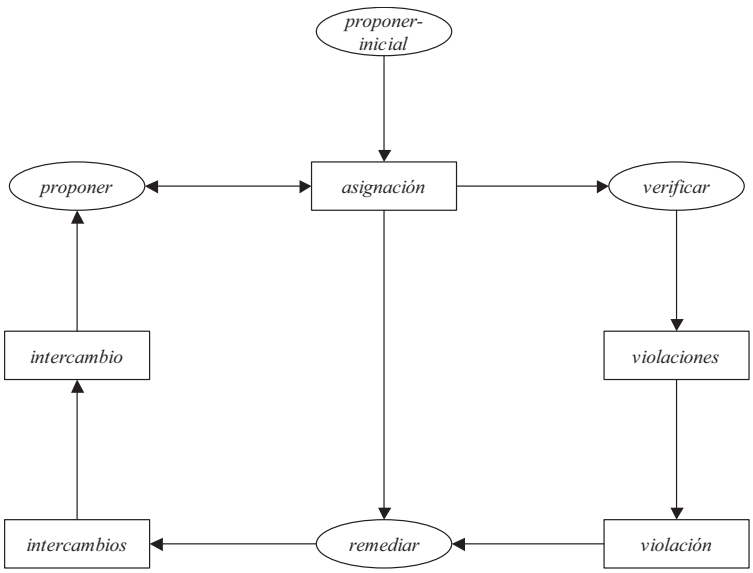


Figura 5.12: Estructura de inferencia del ejemplo de algoritmo 2.

Un ejemplo de sistema que utiliza una estrategia similar a proponer-e-intercambiar es la herramienta general COKE [Poeck, 90; Puppe, 93] que fue aplicada, por ejemplo, al problema de asignación de horarios.

```
METODO proponer-e-intercambiar
  DATOS: ϕ
  RESULTADOS: asignación, éxito

ALGORITMO
1. asignación := ϕ
2. intercambio := ϕ
3. proponer-inicial(-> asignación)
4. verificar(asignación -> violaciones)
5. IF violaciones = ϕ
6. THEN éxito := VERDADERO
7. ELSE
8.   GET(violación, violaciones)
9.   resolver-violación(asignación, violación -> asignación, éxito)

PROCEDIMIENTO generar-propuesta
  DATOS: asignación, intercambio RESULTADOS: asignación, éxito
1. proponer(asignación, intercambio -> asignación)
2. verificar(asignación -> violaciones)
3. IF violaciones = ϕ
4. THEN éxito := VERDADERO
5. ELSE
6    GET(violación, violaciones)
7.   resolver-violación(asignación, violación -> asignación, éxito)

PROCEDIMIENTO resolver-violación
  DATOS: asignación, violación RESULTADOS: asignación, éxito
1. remediar(violación, asignación -> intercambios)
2. IF intercambios = ϕ
3. THEN éxito := FALSO
4. ELSE
5.   REPEAT
6.     GET(intercambio, intercambios)
7.     generar-propuesta(asignación, intercambio -> asignación, éxito)
8.   UNTIL (éxito = VERDADERO) OR (intercambios = ϕ)
```

Figura 5.13: Ejemplo de algoritmo del método de proponer e intercambiar.

5.4 Ejemplos

5.4.1. Ejemplo 1: Ejemplo de aproximación

Para ilustrar la ejecución del método descrito considérese el siguiente ejemplo abstracto que tiene como parámetros A, B,..., G y H en donde los parámetros A, B, C, D y E pueden tener como valor un número entero y el resto, F, G y H, pueden tener como valor alguno del conjunto {a, b, c}. La base de conocimiento de cálculo de parámetros está representada con el siguiente conjunto de reglas:

C1: A = *x* y C = *y* → B = 2**x* + *y*

C2: B > 5 → G = a

C3: B =< 5 → G = b

C4: B > 10, E = *x* → D = 3**x*

C5: B =< 10 → D = 2

C6: E > 10 y H = a → F = b

C7: E =< 10 y H = a → F = a

C8: H = b → F = c

La base de conocimiento de restricciones de diseño es la siguiente:

R1: B =< 13

R2: diferente(B,D)

R3: diferente(G,F)

R4: D < 10

El conocimiento de restricciones de diseño y preferencias de modificaciones se expresa en el siguiente conjunto de sentencias:

```
M1: VIOLACIÓN(R2) -> INCREMENTAR(A,1)
M2: VIOLACIÓN(R2) -> INCREMENTAR(C,1)
M3: VIOLACIÓN(R3) -> SIGUIENTE(H)
M4: VIOLACIÓN(R3) -> DECREMENTAR(E,1)
M5: VIOLACIÓN(R4) -> DECREMENTAR(E,2)
```

En dicho conjunto, por ejemplo, la primera sentencia indica que si no se satisface la restricción R2 una posible solución, es incrementar en una unidad el parámetro A. La tercera sentencia expresa que si no se satisface la restricción R3, una posible solución es cambiar el valor del parámetro H por su siguiente valor, dentro de la lista de posibles valores {a, b, c}. Las medidas de preferencia para remedios son las siguientes:

```
P1: PRIORIDAD(INCREMENTAR(A,X),1)
P2: PRIORIDAD(INCREMENTAR(C,X),2)
P3: PRIORIDAD(SIGUIENTE(H),2)
P4: PRIORIDAD(DECREMENTAR(E,X),4)
```

En un problema concreto, se cuenta con el siguiente conjunto de valores: A=5, C=2, E=4 y H=a. A continuación se muestra el proceso de resolución del problema (se muestra el orden de llamadas a los pasos de inferencia indicando los cambios que se producen en los conjuntos que maneja el algoritmo):

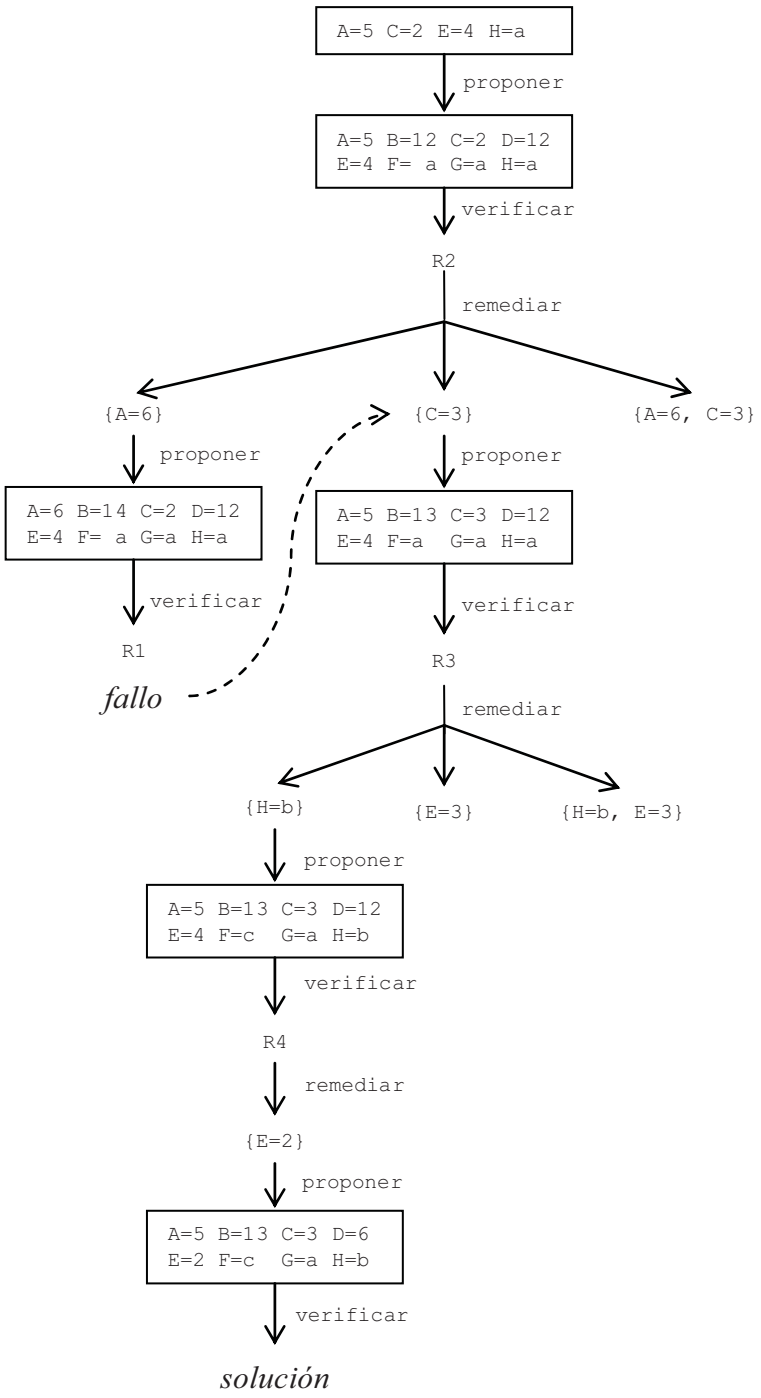


Figura 5.14: Árbol de búsqueda desarrollado en el proceso de resolución del problema.

```
especificaciones = {A=5, C=2, E=4, H=a}
cambio = {}
diseño = {A=5, C=2, E=4, H=a}
1. proponer(diseño, cambio -> diseño)
   diseño = {A=5, B=12, C=2, D=12, E=4, F= a, G=a, H=a}
2. verificar(diseño -> violaciones)
   violaciones = {R2, R3, R4}
   violación = R2
3. remediar(violación, diseño -> cambios)
   cambios = {{A=6}, {C=3}, {A=6, C=3}}
   cambio = {A=6}
4. proponer(diseño, cambio -> diseño)
   diseño = {A=6, B=14, C=2, D=12, E=4, F=a, G=a, H=a}
5. verificar(diseño -> violaciones)
   violaciones = {R1, R3, R4}
   violación = R1
6. remediar(violación, diseño -> cambios)
   cambios = {}
   éxito = FALSO
   se elige otra alternativa de cambios pendiente en paso 3
   cambio = {C=3}
7. proponer(diseño, cambio -> diseño)
   diseño = {A=5, B=13, C=3, D=12, E=4, F=a, G=a, H=a}
8. verificar(diseño -> violaciones)
   violaciones = {R3, R4}
   violación = R3
9. remediar(violación, diseño -> cambios)
   cambios = {{H=b}, {E=3}, {H=b, E=3}}
   cambio = {H=b}
10.proponer(diseño, cambio -> diseño)
```

```
   diseño = {A=5, B=13, C=3, D=12, E=4, F=c, G=a, H=b}
11.verificar(diseño -> violaciones)
   violaciones = {R4}
   violación = R4
12.remediar(violación, diseño -> cambios)
   cambios = {{E=2}}
   cambio = {E=2}
13.proponer(diseño, cambio -> diseño)
   diseño = {A=5, B=13, C=3,D=6, E=2, F=c, G=a, H=b}
14.verificar(diseño -> violaciones)
   violaciones = {}
   éxito = VERDADERO
   se satisfacen todas las restricciones, FIN con éxito
```

La figura 5.14 muestra el árbol de búsqueda desarrollado en el proceso de resolución del problema. El resultado del diseño es:

```
diseño = {A=5, B=13, C=3,D=6, E=2, F= c, G=a, H=b}
```

5.4.2 Ejemplo 2: Diseño del sistema mecánico de un ascensor

Se desarrolla aquí un caso simplificado de diseño del sistema mecánico de un ascensor inspirado en el sistema VT. Se dispone de los parámetros que muestra la figura 5.15.

parámetro		valores posibles
concepto	**atributo**	
edificio	plantas	número entero
cabina	capacidad	número entero
	velocidad	número entero
	aceleración	{suave, normal}
	seguridad	{normal, alta}
	modelo	{a1, a2, a3}
	peso	número entero
cable	modelo	{c1, c2, c3 }
	longitud	número entero
	peso	número entero
motor	tracción	número entero
	máxima-tracción	número entero
contrapeso	peso	número entero
	mínimo-peso	número entero
carga-suspendida	peso	número entero

Figura 5.15: Conjunto de parámetros utilizados en el ejemplo.

Para llevar a cabo el diseño utiliza el método *proponer-y-revisar* con el siguiente conocimiento relativo a cálculo de parámetros:

```
C1:  plantas(edificio) = X → longitud(cable) = 4*X
C2:  capacidad(cabina) < 5 → modelo(cabina) = a1
C3:  capacidad(cabina) = [5,7] → modelo(cabina) = a2
C4:  capacidad(cabina) > 7 → modelo(cabina) = a3
C5:  modelo(cabina) = a1 → peso(cabina) = 200
C6:  modelo(cabina) = a2 → peso(cabina) = 250
C7:  modelo(cabina) = a3 → peso(cabina) = 300
```

C8: peso(cabina) = X → peso(contrapeso) = X/5

C9: seguridad(cabina) = normal → modelo(cable) = c1

C10: seguridad(cabina) = alta → modelo(cable) = c2

C11: modelo(cable) = c1 y longitud(cable) = X → peso(cable) = 2*X

C12: modelo(cable) = c2 y longitud(cable) = X → peso(cable) = 3*X

C13: modelo(cable) = c3 y longitud(cable) = X → peso(cable) = 4*X

C14: peso(cabina) = X y peso(cable)=Y y peso(contrapeso) = Z
→ peso(carga-suspendida) = X + Y + Z

C15: peso(carga-suspendida) = X y velocidad(cabina) = Y
→ tracción(motor) = X * Y

C16: modelo(cable) = c1 → máxima-tracción(motor) = 5000

C17: modelo(cable) = c2 o c3 → máxima-tracción(motor) = 2500

C18: tracción(motor) >= 2000 → mínimo-peso(contrapeso) = 30

C19: tracción(motor) < 2000 y aceleración(cabina) = suave
→ mínimo-peso(contrapeso) = 60

C20: tracción(motor) < 2000 y aceleración(cabina) = normal
→ mínimo-peso(contrapeso) = 45

Se dispone también de las siguientes restricciones de diseño:

R1: longitud(cable) < 100

R2: peso(contrapeso) > mínimo-peso(contrapeso)

R3: peso(carga-suspendida) < 1000

R4: tracción(motor) < máxima-tracción(motor)

R5: peso(cabina)/peso(carga-suspendida) < 0.65

R6: | 23 - tracción(motor)/100 | > 1

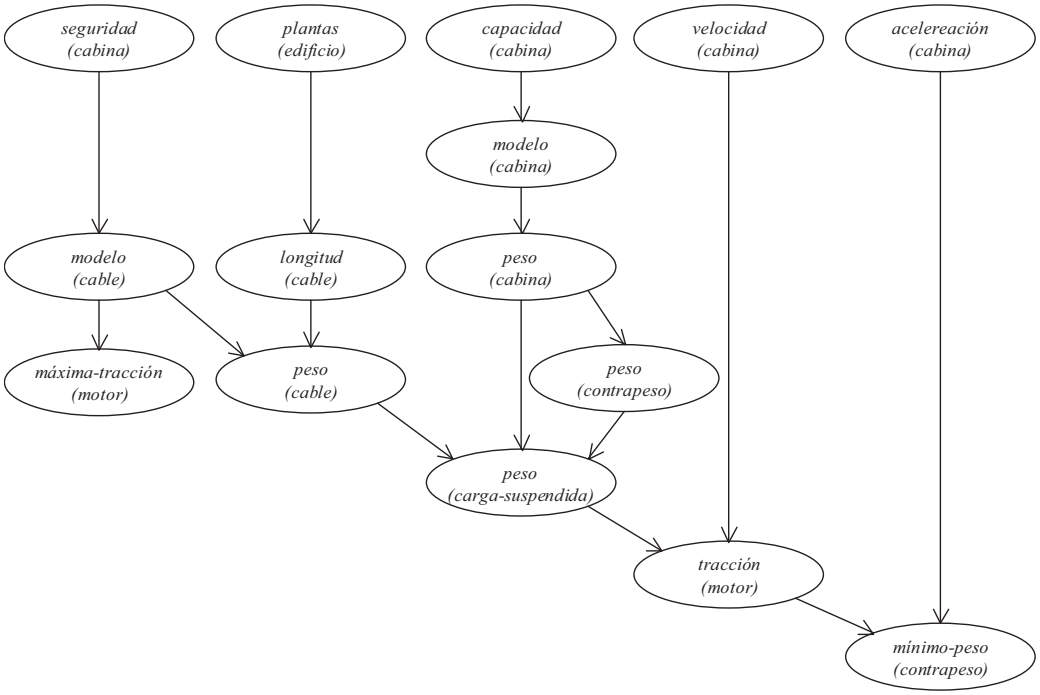


Figura 5.16: Red de cálculos formada por los parámetros del problema.

Respecto a modificaciones se sabe lo siguiente:

M1: VIOLACIÓN(R2) → SIGUIENTE(modelo(cabina))

M2: VIOLACIÓN(R2) → ANTERIOR(aceleración(cabina))

M3: VIOLACIÓN(R3) → ANTERIOR(modelo(cabina))

M4: VIOLACIÓN(R3) → ANTERIOR(modelo(cable))

M5: VIOLACIÓN(R3) → ANTERIOR(seguridad(cabina))

M6: VIOLACIÓN(R4) → ANTERIOR(modelo(cabina))

M7: VIOLACIÓN(R4) → DECREMENTAR(velocidad(cabina),2)

M8: VIOLACIÓN(R5) → ANTERIOR(modelo(cabina))

M9: VIOLACIÓN(R5) → SIGUIENTE(modelo(cable))

Las prioridades de modificaciones son las siguientes:

```
P1: PRIORIDAD(ANTERIOR(modelo(cabina),X),1)
P2: PRIORIDAD(SIGUIENTE(modelo(cabina),X),3)
P3: PRIORIDAD(ANTERIOR(modelo(cable)),X),2)
P4: PRIORIDAD(SIGUIENTE(modelo(cable)),X),2)
P5: PRIORIDAD(DECREMENTAR(velocidad(cabina)),X),2)
P6: PRIORIDAD(ANTERIOR(aceleración(cabina)),4)
P7: PRIORIDAD(ANTERIOR(seguridad(cabina)),5)
```

En un caso concreto de diseño se desea configurar la maquinaria de un ascensor para un edificio de 10 plantas, que sea capaz de transportar 4 personas, con un valor de 6 para velocidad de ascenso, nivel de seguridad normal y aceleración normal. A partir dichos datos, aplica el método *proponer-y-revisar* para encontrar un conjunto de valores de parámetros de diseño que sean compatibles con el conocimiento de diseño expresado. Seguidamente se describe el proceso de resolución del problema:

```
especificaciones = {plantas(edificio)=10, capacidad(cabina)=4,
   velocidad(cabina)=6, seguridad(cabina)=normal,
   aceleración(cabina)=normal}
cambio = {}
diseño = {plantas(edificio)=10, capacidad(cabina)=4,
   velocidad(cabina)=6, seguridad(cabina)=normal,
   aceleración(cabina)=normal}
1. proponer(diseño, cambio -> diseño)
  diseño = {plantas(edificio)=10, capacidad(cabina)=4,
     velocidad(cabina)=6,seguridad(cabina)=normal,
     aceleración(cabina)=normal, longitud(cable)=40,
     modelo(cabina)=a1, peso(cabina)=200,
     peso(contrapeso)=40, modelo(cable)=c1 ,peso(cable)=80,
```

```
     peso(carga-suspendida)=320, tracción(motor)=1920,
     máxima-tracción(motor)=5000, mínimo-peso(contrapeso)=45}
  (se subrayan los parámetros que cambian el valor)
2. verificar(diseño -> violaciones))
   violación = R2
3. remediar(violación, diseño -> cambios)
   para resolver la violación se puede hacer:
      modelo(cabina)=a2 (prioridad 3)
      aceleración(cabina)=suave (prioridad 4)
   cambios = {{modelo(cabina)=a2}, {aceleración(cabina)=suave},
              {modelo(cabina)=a2, aceleración(cabina)=suave}}
   cambio = {modelo(cabina)=a2}
4. proponer(diseño, cambio -> diseño)
   diseño = {plantas(edificio)=10, capacidad(cabina)=4,
     velocidad(cabina)=6, seguridad(cabina)=normal,
     aceleración(cabina)=normal, longitud(cable)= 40,
     modelo(cabina)=a2, peso(cabina)=250,
     peso(contrapeso)=50, modelo(cable)=c1, peso(cable)=80,
     peso(carga-suspendida)=380, tracción(motor)=2280,
     máxima-tracción(motor)=5000, mínimo-peso(contrapeso)=30}
5. verificar(diseño -> violaciones))
   violación = R5
6. remediar(violación, diseño -> cambios)
   para resolver la violación se puede hacer:
      modelo(cabina)=a1 (prioridad 1)
      modelo(cable)=c2 (prioridad 2)
   cambios = {{modelo(cabina)=a1}, {modelo(cable)=c2},
              {modelo(cabina)=a1, modelo(cable)=c2}}
   cambio = {modelo(cabina)=a1}
7. proponer(diseño, cambio -> diseño)
```

```
   diseño = {plantas(edificio)=10, capacidad(cabina)=4,
     velocidad(cabina)=6, seguridad(cabina)=normal,
     aceleración(cabina)=normal, longitud(cable)=40,
     modelo(cabina)=a1, peso(cabina)=200,
     peso(contrapeso)=40, modelo(cable)=c1 ,peso(cable)=80,
     peso(carga-suspendida)=320, tracción(motor)=1920,
     máxima-tracción(motor)=5000, mínimo-peso(contrapeso)=45}
8. verificar(diseño -> violaciones))
   es un estado repetido
   se da FALLO por esta rama
   se vuelve a otra alternativa del paso 6
9. proponer(diseño, cambio -> diseño)
   diseño = {plantas(edificio)=10, capacidad(cabina)=4,
     velocidad(cabina)=6, seguridad(cabina)=normal,
     aceleración(cabina)=normal, longitud(cable)= 40,
     modelo(cabina)=a2, peso(cabina)=250,
     peso(contrapeso)= 50, modelo(cable)=c2, peso(cable)=120,
     peso(carga-suspendida)=420, tracción(motor)=2520,
     máxima-tracción(motor)=2500, mínimo-peso(contrapeso)=30}
10.verificar(diseño -> violaciones))
   violación = R4
11.remediar(violación, diseño -> cambios)
   para resolver la violación se puede hacer:
      modelo(cabina)=a1 (prioridad 1)
      velocidad(cabina)=4 (prioridad 4)
   la modificación ANTERIOR(seguridad(cabina)) no es aplicable
   porque seguridad(cabina)=normal es el valor más bajo
   cambios = {{modelo(cabina)=a1}, {velocidad(cabina)=4},
              {modelo(cabina)=a1, velocidad(cabina)=4}}
   cambio = {modelo(cabina)=a1}
```

```
12.proponer(diseño, cambio -> diseño)
   diseño = {plantas(edificio)=10, capacidad(cabina)=4,
      velocidad(cabina)=6, seguridad(cabina)=normal,
      aceleración(cabina)=normal, longitud(cable)= 40,
      modelo(cabina)=a1, peso(cabina)=200,
      peso(contrapeso)= 40, modelo(cable)=c2 ,peso(cable)= 120,
      peso(carga-suspendida)= 360, tracción(motor)= 2160,
      máxima-tracción(motor) = 2500, mínimo-peso(contrapeso)= 30}
13.verificar(diseño -> violaciones)
   violaciones = {}
   éxito = VERDADERO
   se satisfacen todas las restricciones, FIN con éxito
```

Para obtener una solución coherente con las restricciones de diseño no ha sido necesario modificar los valores de especificación. La solución final es la siguiente:

```
   diseño = {plantas(edificio)=10, capacidad(cabina)=4,
      velocidad(cabina)=6, seguridad(cabina)=normal,
      aceleración(cabina)=normal, longitud(cable)= 40,
      modelo(cabina)=a1, peso(cabina)=200,
      peso(contrapeso)= 40, modelo(cable)=c2, peso(cable)= 120,
      peso(carga-suspendida)= 360, tracción(motor)= 2160,
      máxima-tracción(motor) = 2500, mínimo-peso(contrapeso)= 30}
```

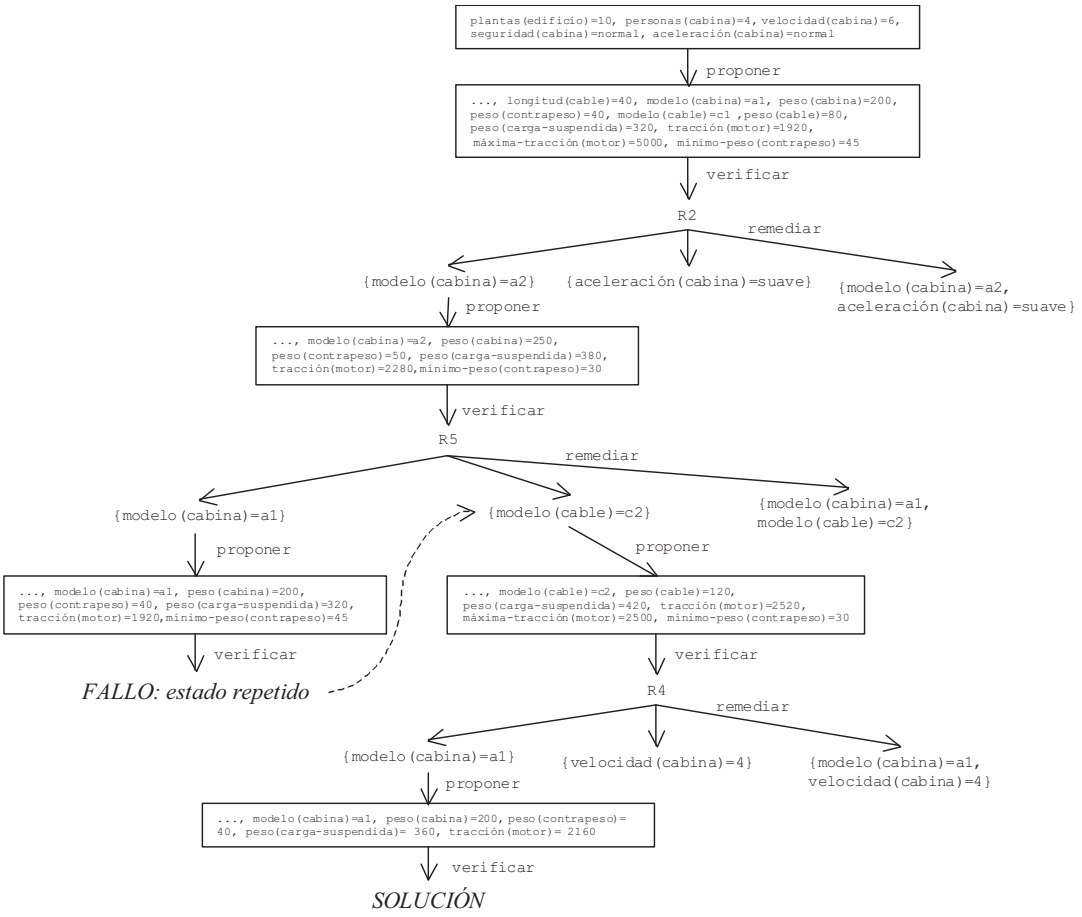


Figura 5.17: Árbol desarrollado para la búsqueda de la solución en el ejemplo.

5.4.3 Ejemplo 3: Control de tráfico en autovías urbanas

En este apartado se muestra la aplicación del método de proponer y revisar a un caso de control de tráfico en carreteras. El ejemplo corresponde a un sistema real denominado TRYS (Tráfico: Razonamiento y Simulación) que fue desarrollado para facilitar la señalización automática en autovías urbanas.

La tarea de gestionar el tráfico en áreas conflictivas, tales como entradas y salidas a grandes ciudades o cinturones de circunvalación, supone la necesidad de manejar en cortos periodos de tiempo gran cantidad de información, así como tener una experiencia notable no sólo en el control de tráfico en general, sino sobre el comportamiento concreto del mismo. El sistema TRYS [Molina et al., 98; Molina, Robledo, 01] tiene como fin principal asistir a los operadores de un centro de control de tráfico en la decisión sobre acciones de señalización de acuerdo con la presencia de problemas de tráfico.

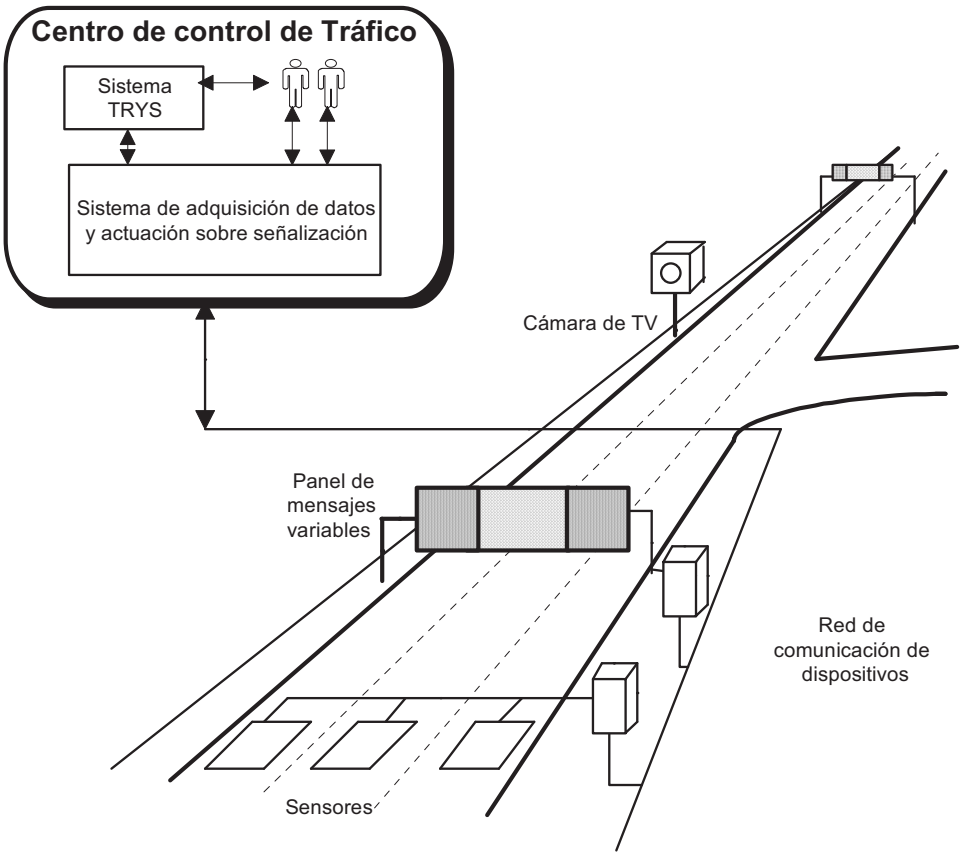


Figura 5.18: Contexto de operación del sistema TRYS

TRYS hace uso de los datos medidos por sensores que proporcionan magnitudes básicas del tráfico tales como la intensidad, ocupación y velocidad, etc. aprovechando una infraestructura de sensores y comunicaciones instaladas en las principales carreteras de las ciudades (figura 5.18). El sistema TRYS periódicamente analiza datos obtenidos en tiempo real y, como resultado de su razonamiento, realiza una propuesta de señalización para los dispositivos de control la cual, bajo confirmación del operador, puede ser difundida a carretera. Las propuestas incluyen, por ejemplo, mensajes sobre presencia de retenciones, mensajes sobre presencia de incidentes, mensajes sobre tiempos de recorrido hasta un punto destino, recomendación de itinerarios, etc.

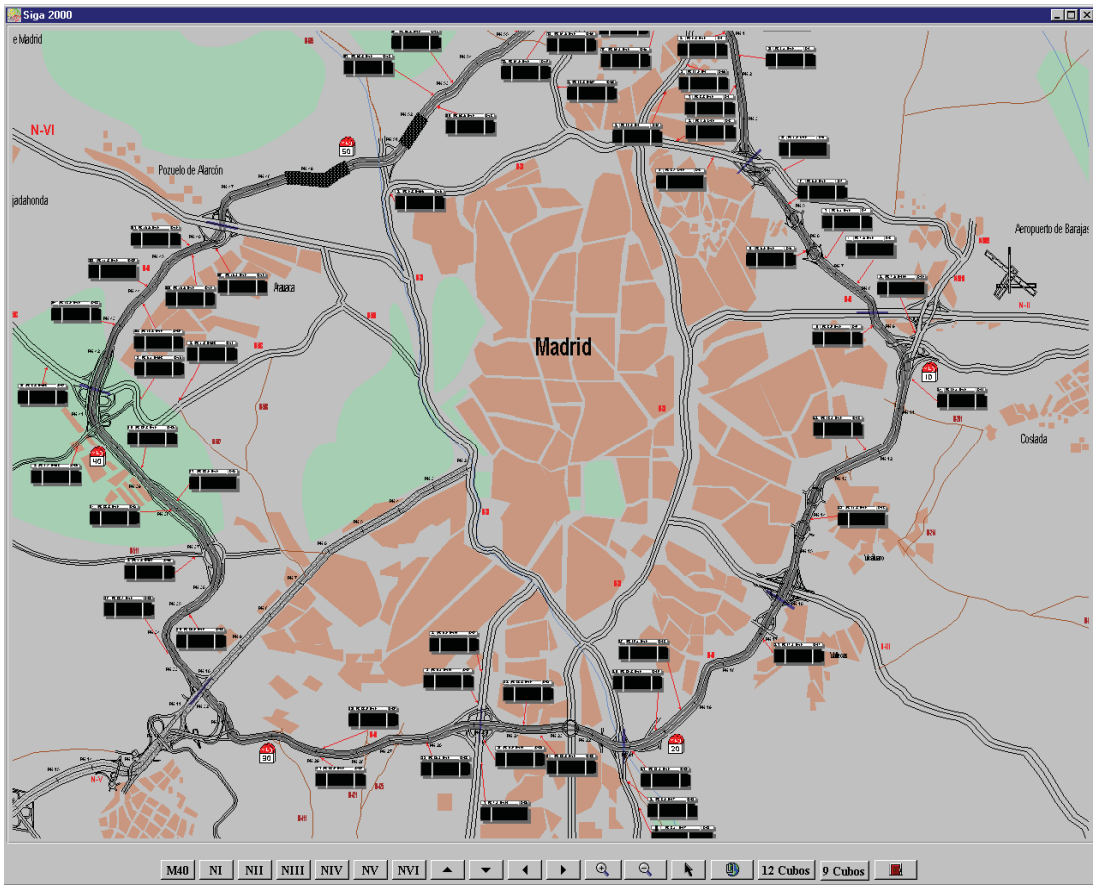


Figura 5.19: Ejemplo de conjunto de dispositivos de señalización en la ciudad de Madrid

En la construcción del sistema TRYS se ha hecho uso del método de proponer y revisar que facilitó significativamente la adquisición y organización del conocimiento. En la figura 5.20 se muestra la organización genérica seguida, en donde se muestra las diversas bases de conocimiento.

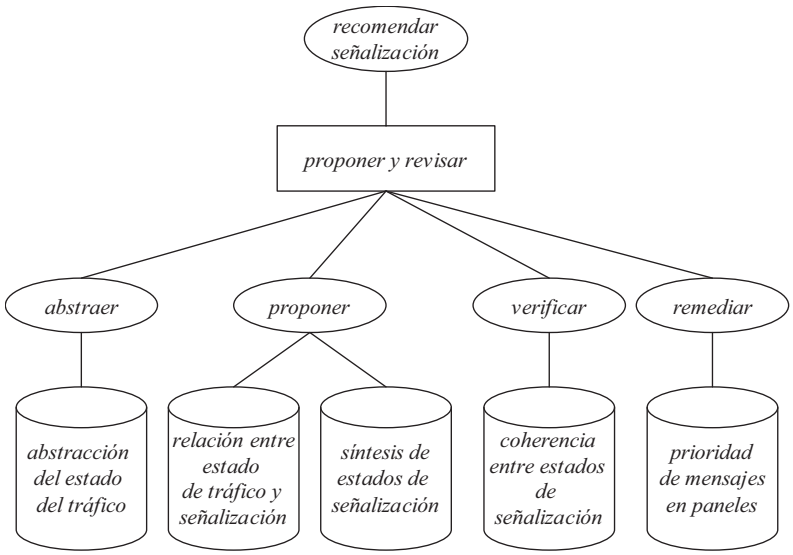


Figura 5.20: Organización del conocimiento del sistema TRYS

El sistema cuenta con cinco tipos de bases de conocimiento: (1) *base de abstracción del estado del tráfico*, que contiene los criterios propios del centro de control necesarios para determinar el estado de las diversas zonas de la carretera (por ejemplo, en esta base incluye criterios de interpretación de datos para determinar la presencia de retenciones o métodos locales de estimación de tiempos de recorrido), (2) *base de relación entre el estado de tráfico y los estados de señalización*, que recoge afirmaciones sobre el estado de tráfico que debe darse para que un determinado estado de señalización se pueda presentar a los conductores, (3) *base de síntesis de estados de señalización*, que contiene criterios que permiten sintetizar varios estados de señalización diferentes para un mismo dispositivo en un único estado, como por ejemplo sintetizar tres mensajes de tiempos de recorrido en un

único mensaje que señaliza los tiempos a los tres lugares distintos, (4) *base de coherencia entre estados de señalización,* que incluye afirmaciones sobre incompatibilidad entre grupos de estados de señalización, lo que permite al sistema detectar y rechazar situaciones incoherentes en la señalización de carretera y (5) *base de prioridad de mensajes en paneles*.

Como se observa en la figura 5.20, el conocimiento de las bases (2) y (3) se utiliza para realizar la fase de propuesta de mensajes en paneles, el conocimiento de la base (4) se maneja para verificar si la propuesta es coherente y el conocimiento de la base (5) se utiliza con el fin de encontrar opciones alternativas cuando se detectan incoherencias. Además, se ha realizado una extensión del método proponer y revisar incluyendo un paso previo adicional para realizar una fase de abstracción que utiliza la base (1). El proceso de razonamiento del sistema desarrolla un proceso iterativo de generación de combinaciones y de estudio de coherencia que se apoya en el conocimiento sobre prioridades de mensajes de paneles. Este procedimiento permite manejar el hecho de que haya varias propuestas de mensajes candidatos para cada panel y que ciertas combinaciones sean incompatibles. El resultado final generado por el sistema es la propuesta de cambios en la señalización actual, que se obtiene mediante comparación de la propuesta generada y el estado actual de señalización.

Para representar el conocimiento, TRYS maneja un lenguaje simbólico, fundamentalmente basado en reglas, que recoge de forma explícita afirmaciones individuales sobre formas de señalizar, expresando de forma condicional relaciones entre estados de tráfico y mensajes de paneles, además de relaciones entre estado de tráfico y nueva información inferida acerca de la vía. El lenguaje de reglas utilizado aporta una elevada flexibilidad para formular una gran variedad de condiciones sobre el estado del tráfico, permitiendo incluir desde afirmaciones simples sobre medidas obtenidas directamente por sensores hasta descripciones más complejas que recogen ciertos algoritmos más elaborados para decidir sobre la conveniencia de un mensaje. Esta flexibilidad es especialmente útil en control de tráfico en donde

en gran número de ocasiones los criterios de control no pueden ser expresados de forma general para todos los casos (lo que impide hacer uso de soluciones algorítmicas genéricas) sino que deben ser formulados uno a uno. El manejo de bases de conocimiento permite formular cada estrategia a la medida de las necesidades de cada lugar con el suficiente grado de adaptación y especificidad.

El sistema TRYS cuenta además con una herramienta informática que ayuda a la construcción de modelos (figura 5.21). Dicha herramienta está formada por varios editores que guían al usuario desarrollador en la formulación de los criterios de señalización de la carretera en función de los problemas considerados mostrando diferentes visiones del modelo, en un lenguaje cercano al dominio profesional de la gestión de tráfico y asegurando la coherencia de los diferentes criterios que se escriben.

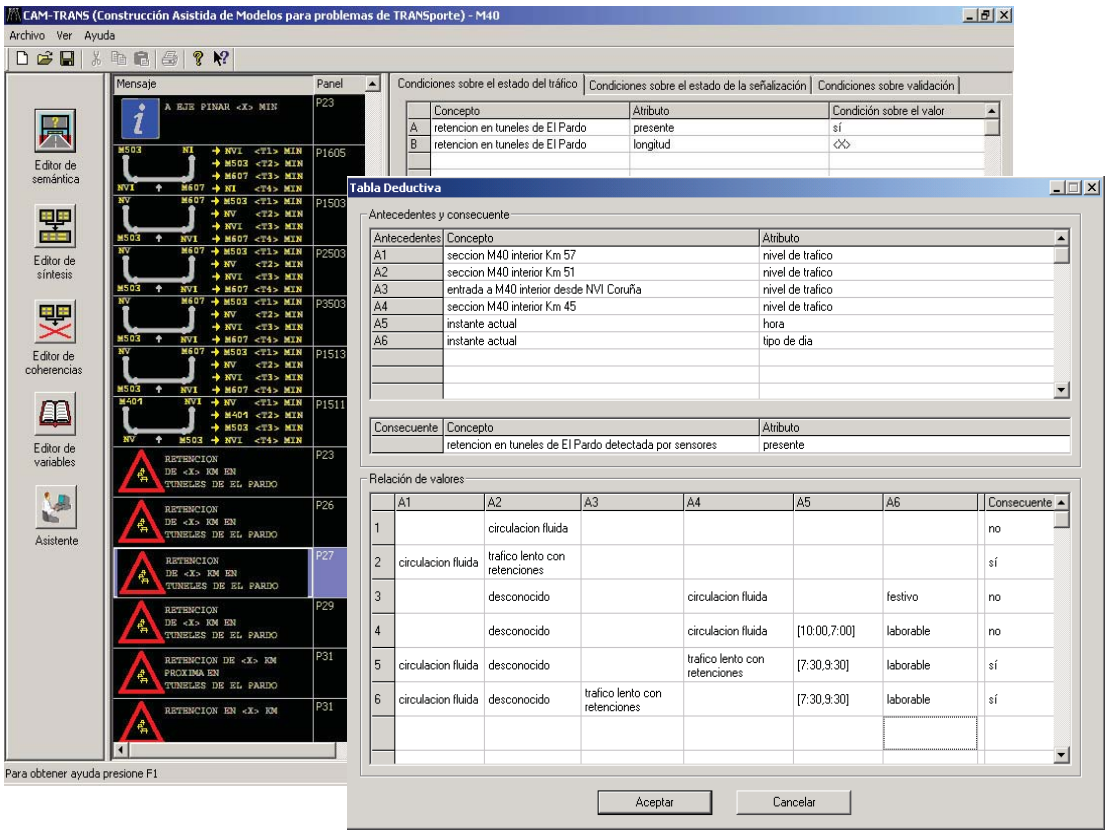


Figura 5.21: Interfaz de usuario de la herramienta de construcción de modelos TRYS.

5.5 Ejercicios

EJERCICIO 5.1 Considérese un modelo diseñado para resolver un problema de configuración que maneja los siguientes parámetros:

Parámetro	Valores Posibles
P1	{bajo,medio,alto}
P2	número
P3	número
P4	número
P5	número
P6	número
P7	número
P8	{decreciente, creciente}
P9	número

El modelo sigue el método *proponer-y-revisar* y maneja siguiente conocimiento relativo a cálculo de parámetros:

```
C1: P1=alto → P4=100
C2: P1=medio → P4=200
C3: P1=bajo → P4=300
C4: P1=alto o P1=medio → P5 = P2
C5: P1 =bajo → P5 = 0
C6: P6 = P2 + P3
C7: P7 = P4 / 100
C8: P5 > 50 → P8 = creciente
C9: P5 =< 50 → P8 = decreciente
C10: P9 = P5 * P6
```

Respecto a restricciones de configuración se tienen las siguientes sentencias (el símbolo =/= significa *distinto de*):

```
R1: P2 = P3 → P9 >= 60
R2: P6 < 18
R3: P9 = 0 → P8 =/= decreciente
R4: P5 < P7 + P9
R5: P9 < 120
R6: P3 = 0 → P6 =< 6
```

Finalmente, sobre conocimiento de remedios se tienen las siguientes sentencias:

```
M1: VIOLACION(R1) → INCREMENTAR(P2,5)
M2: VIOLACION(R3) → SIGUIENTE(P1)
M3: VIOLACION(R3) → INCREMENTAR(P2,10)
M4: VIOLACION(R4) → SIGUIENTE(P1)
M5: VIOLACION(R4) → DECREMENTAR(P2,7)
M6: VIOLACION(R5) → DECREMENTAR(P3,5)
M7: VIOLACION(R5) → DECREMENTAR(P2,5)
```

Por ejemplo, la sentencia M1 indica que, en caso de no cumplirse la restricción R1, una posible solución sería incrementar el valor del parámetro P2 en 5 unidades. La sentencia M2 indica que, en caso de no cumplirse la restricción R3, una posible solución sería asignar al parámetro P1 el siguente valor cualitativo, de acuerdo con el conjunto de valores posibles que dicho parámetro tiene.

SE PIDE:

Aplicar el método *proponer-y-revisar* para obtener una configuración de valores de parámetros coherente con el conocimiento anterior, sabiendo que como valores iniciales de parámetros se tiene P1= bajo, P2=10 y P3=5. Como estrategia de

control, si es necesario elegir entre dos opciones, elegir según el orden que presentan en el enunciado. Dibujar el árbol correspondiente al proceso de resolución.

EJERCICIO 5.2. Se tiene un modelo basado en el método *proponer-y-revisar* sobre el que se dispone del siguiente conocimiento relativo a cálculo de parámetros expresado en forma de reglas:

C1: A = *x* y B = *y* → D = *x* + *y*

C2: B = *x* y C = *y* → E = *x* − *y*

C3: C = *x* → F = *x*/2

C4: D = *x* y E = *y* → G = *x* * *y*

C5: E = *x* y F = *y* → H = *x* + *y*

C6: E = *x* → I = 2**x*

Sobre restricciones de configuración se tienen las siguientes sentencias (el símbolo =/= significa *distinto de*):

R1: G =/=55

R2: E + F >= 6

R3: H =/= 7.5

Finalmente, sobre conocimiento de remedios se tienen las siguientes sentencias:

M1: VIOLACIÓN(R1) → DECREMENTAR(B,2)

M2: VIOLACIÓN(R1) → DECREMENTAR(A,1)

M3: VIOLACIÓN(R3) → DECREMENTAR(C,1)

Por ejemplo, la primera sentencia M1 indica que, en caso de no cumplirse la restricción R1, una posible solución es decrementar el parámetro B en 2 unidades.

SE PIDE:
Aplicar el método *proponer-y-revisar* para obtener una configuración de valores de parámetros coherente con el conocimiento anterior, sabiendo que como valores iniciales de parámetros se tiene A=1, B=10 y C=5. Como estrategia de control, si es necesario elegir entre dos opciones, elegir según el orden que presentan en el enunciado. Dibujar el árbol correspondiente al proceso de resolución.

EJERCICIO 5.3. Se dispone de un modelo basado en el método *proponer-y-revisar* del que se tiene el siguiente conocimiento relativo a cálculo de parámetros formulado como expresiones aritméticas:

```
C1: D = A · B
C2: E = B + C
C3: F = 2 · D
C4: G = D − E
C5: H = E / 2
```

Respecto a restricciones de configuración se tienen las siguientes sentencias (el símbolo =/= significa *distinto de*):

```
R1: F =/= 30
R2: E + F >= 20
R3: D − H =/= 4
R4: H =/=  6.5
R5: F + G + H < 50
```

Finalmente, sobre conocimiento de remedios se tienen las siguientes sentencias:

```
M1: VIOLACIÓN(R1) → DECREMENTAR(A,1)
M2: VIOLACIÓN(R1) → INCREMENTAR(B,1)
M3: VIOLACIÓN(R2) → DECREMENTAR(C,5)
M4: VIOLACIÓN(R2) → INCREMENTAR(B,2)
M5: VIOLACIÓN(R4) → INCREMENTAR(C,1)
M6: VIOLACIÓN(R4) → DECREMENTAR(B,1)
```

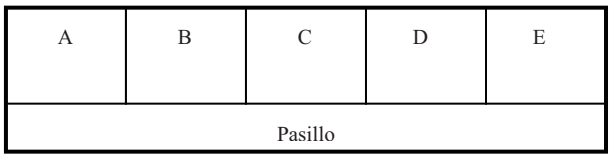
Por ejemplo, la primera sentencia M1 indica que, en caso de no cumplirse la restricción R1, una posible solución es decrementar el parámetro A en una unidad.

SE PIDE:

1. Aplicar el método *proponer-y-revisar* para obtener una configuración de valores de parámetros coherente con el conocimiento anterior, sabiendo que como valores iniciales de parámetros se tiene A=3, B=5 y C=7. Como estrategia de control, si es necesario elegir entre dos opciones, elegir según el orden que presentan en el enunciado. Dibujar el árbol correspondiente al proceso de resolución.

2. Explicar cómo debería modificarse el modelo si se quisiera indicar que el valor del parámetro D se obtiene también como suma de otros dos parámetros tal como indica la siguiente sentencia: D = F + G.

EJERCICIO 5.4. Una empresa de informática desea trasladar a uno de sus equipos de desarrollo de sistemas a un nuevo edificio. Dicho equipo está formado por un jefe de proyecto J, una secretaria S, dos analistas A1, A2 y cuatro programadores P1, P2, P3 y P4. La distribución de las oficinas en donde se deben instalar es la siguiente:



El problema que se plantea es encontrar una asignación de personas a las oficinas siguiendo una serie de criterios orientados a hacer un buen uso del espacio disponible y, además, tener en cuenta las preferencias del personal. En principio, como estrategia general se tratará de ubicar a las personas en la oficina A o en oficinas lo más cercanas a ésta, sabiendo que la capacidad máxima de cada oficina es de 4 personas. Además se deberán cumplir las siguientes restricciones:

- El jefe de proyecto debe estar solo en una oficina o con la secretaria
- La secretaria debe estar con el jefe de proyecto o en la sala contigua
- No puede haber en la misma sala fumadores y no fumadores (J, S, P1 y P2 son fumadores)
- Los analistas no pueden estar juntos en la misma oficina.
- Los analistas deben estar próximos al jefe, es decir, no debe haber más de una oficina intermedia que los separe.
- En la oficina B no puede haber más de 2 personas.

Si durante el proceso de resolución de este problema se viola alguna de estas restricciones, la forma de intentar resolver dicha violación será reasignando una o varias personas a otras oficinas. Entre varias formas posibles de resolver una violación tendrán más prioridad las asignaciones a oficinas más cercanas a la que inicialmente se había considerado y las que supongan un menor cambio en el número de personas.

SE PIDE:

1. Explicar cómo se puede representar este problema de asignación para resolverlo mediante el método de *proponer-e-intercambiar*. Mostrar los parámetros considerados y el contenido de cada una de las bases de conocimiento.

2. Aplicar el método de *proponer-e-intercambiar* para encontrar una asignación posible de personas que sea compatible con los requisitos indicados. Indicar de forma detallada los pasos realizados y dibujar el árbol de resolución del problema en donde se muestren las ramas correspondientes a las diferentes opciones de búsqueda.

EJERCICIO 5.5. En un centro hospitalario se desea planificar las guardias del servicio de urgencias para el mes que se indica en la figura. Para ello se cuenta con una plantilla de 6 médicos adjuntos a los que se denominará A, B, C, D, E y F, además de 7 médicos residentes a los que se les denominará M, N, O, P, Q, R, S. Para cada día es necesaria la presencia de un médico adjunto y un médico residente.

L	M	X	J	V	S	D
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Para encontrar una solución, el problema se divide en dos partes. Primero se busca una solución para los médicos adjuntos y después otra para los médicos residentes que sea compatible con la encontrada para los médicos adjuntos. En la primera parte del problema se aplica una adaptación del método *proponer-e-intercambiar* de la siguiente forma. Se cuenta con un conjunto de parámetros A1, A2, ..., A31 en donde cada parámetro corresponde a un día del mes. El dominio de valores de cada parámetro es {A, B, C, D, E, F}, es decir, el conjunto de los médicos adjuntos.

Para obtener la solución se parte de una propuesta inicial que se obtiene de la forma siguiente. A cada día del mes se le asigna provisionalmente un médico de forma cíclica de acuerdo con el orden de los días y siguiendo el orden alfabético de los médicos. Para ello, se parte del primer día del mes con el médico A, el segundo día el médico B, hasta llegar al F en el día sexto y, al séptimo día se asigna otra vez el A, al siguiente el B, etc. repitiendo el proceso hasta completar el mes. Esta propuesta inicial así obtenida debe ser modificada a continuación para cumplir el siguiente conjunto de restricciones:

R1: Un médico no debe realizar guardia dos fines de semana seguidos.
R2: En un mes todos los médicos deben tener al menos una guardia en fin de semana.
R3: Si el médico ha tenido guardia un fin de semana, entonces no puede tener guardia el siguiente lunes.
R4: Un médico no debe realizar guardia dos días seguidos.

Restricción violada	Acciones posibles para remediar la violación de la restricción	Prio-ridad	Notación
R1	intercambiar médico de día Y con médico de viernes anterior a Y	1	X e Y son los días que incumplen R1, X < Y
	intercambiar médico de día Y con médico de jueves anterior a Y	2	
	intercambiar médico de día Y con médico de miércoles anterior a Y	3	
	intercambiar médico de día Y con médico de martes anterior a Y	4	
	intercambiar médico de día Y con médico de lunes anterior a Y	5	
R2	intercambiar médico de día X con médico de primer sábado del mes	1	X es el día de la primera guardia del mes del médico que incumple R2
	intercambiar médico de día X con médico de primer domingo del mes	2	
	intercambiar médico de día X con médico de segundo sábado del mes	3	
	intercambiar médico de día X con médico de segundo domingo del mes	4	
R3	intercambiar médico de día X con médico de siguiente martes	1	X es el lunes que viola R3
	intercambiar médico de día X con médico de siguiente miércoles	2	
R4	intercambiar médico de día X con médico de día anterior a X	1	X e Y son los días que incumplen R4, X < Y
	intercambiar médico de día Y con médico de día posterior a Y	2	

Para remediar las violaciones de restricciones se pueden realizar las acciones que se muestran en la tabla adjunta en donde cada fila indica una acción posible para remediar la violación de una restricción. Cada acción tiene una medida numérica de prioridad, entendiendo como más prioritarios los números más bajos.

En la segunda parte del problema para obtener la asignación de médicos residentes se aplica también *proponer-e-intercambiar* de la forma descrita, pero con los parámetros B1, ..., B31, tantos como días del mes, con el dominio de valores {M, N, O, P, Q, R, S}, tantos como médicos residentes. Se aplican aquí los mismos criterios que antes además de las siguientes restricciones adicionales (para las cuales no se tienen acciones para remediarlas):

R5: el médico adjunto C no debe estar de guardia con el médico residente N los lunes

R6: el médico adjunto A no debe estar de guardia con el médico residente P

SE PIDE:

1. Aplicar el método *proponer-e-intercambiar* para la primera parte del problema para encontrar valores de los parámetros A1, ..., A31. Mostrar únicamente los primeros 10 pasos de inferencia indicando de forma detallada los valores de las variables que se manejan en el algoritmo así como los elementos de las bases de conocimiento utilizados en cada paso de inferencia.

2. Continuar con la resolución de la primera pare del problema para encontrar una asignación a los parámetros A1, ..., A31, dibujando únicamente el árbol de búsqueda e indicar cuál es la primera solución encontrada.

3. Mostrar cuál es la solución de la segunda parte del problema encontrando una solución para los médicos residentes con los parámetros B1, ..., B31. En la resolución de esta parte del problema mostrar únicamente el árbol de búsqueda desarrollado.

NOTA: Durante el proceso de búsqueda, ante varias opciones posibles en donde no se indique otro criterio de control, elegir primero los elementos según el orden en que se presentan en el enunciado.

EJERCICIO 5.6. Considerar el ejemplo 2 sobre diseño mecánico de un ascensor que se describe en el presente capítulo.

SE PIDE:

1. ¿Cuál es el factor de ramificación *FR* del árbol de búsqueda en este problema, para cualquier conjunto de datos de entrada? (El factor de ramificación *FR* es el máximo número de arcos que pueden partir de un nodo hacia el siguiente nivel inferior en la jerarquía). ¿Por qué?

2. Para cada una de las siguientes sentencias que forman parte de las bases de conocimiento del ejemplo 2, indicar de qué tipo de conocimiento se trata (heurístico, profundo, de control, meta-conocimiento, genérico, declarativo, procedimental):

   - C11: “El peso del cable es el doble de su longitud si el modelo de cable es C1”

   - R3: “El peso de la carga suspendida no debe ser superior a 1000”.

- M7: “Para evitar que la tracción supere el valor máximo puede reducirse la velocidad de ascenso con prioridad 4”

3. Se desea modificar el contenido de las bases de conocimiento del ejemplo 2 de diseño mecánico para incluir los criterios que indican las frases que se presentan a continuación. Escribir las sentencias que habría que añadir o modificar en las bases de conocimiento y mostrar, en su caso, qué información adicional sería necesario adquirir de los expertos en el problema:

    - “Si el peso de la carga suspendida es mayor que 200 el modelo de cable es C2 o C3”

    - “La elección del modelo de cable afecta más al diseño que la elección del modelo de cabina”

    - “Si el peso del contrapeso no es suficiente se puede elegir otro modelo de cable más barato u otro de otra firma comercial”

4. Para cada una de las frases anteriores, indicar de qué tipo de conocimiento se trata (profundo, heurístico, de control, meta-conocimiento, genérico).

6 Planificación jerárquica HTN

En este capítulo se presenta una forma de resolver problemas de planificación con un enfoque heurístico y jerárquico que permite dar una solución eficiente a problemas complejos de planificación. El método descrito es aplicable también a problemas de configuración.

El método se basa en realizar un descenso jerárquico dirigido por planes de acciones abstractas que se van refinando o descartando progresivamente según se estudian los detalles en niveles inferiores de la jerarquía. Inicialmente, el método parte de un posible plan abstracto para alcanzar el objetivo y, en sucesivos pasos, las acciones de dicho plan se van detallando buscando nuevos subplanes en un proceso de búsqueda tentativa, que puede replantear pasos previos, hasta alcanzar un plan suficientemente detallado.

El método que se presenta aquí se encuadra dentro de los métodos denominados de *planificación jerárquica en redes de tareas* o planificadores HTN (*Hierarchical Task Network*) que se basan en un enfoque heurístico para dirigir la búsqueda de los componentes de los planes.

6.1 Descripción general

En los problemas de planificación se tiene un sistema sobre el cual se realizan acciones y se busca un plan destinado a modificar su estado de acuerdo con ciertos objetivos. La terminología habitualmente utilizada en problemas de planificación es la siguiente:

- *Estado del sistema*: conjunto de atributos que expresan el estado del sistema en donde actúa el planificador. Cada atributo puede tomar un valor (cualitativo o cuantitativo) dentro de un conjunto de opciones posibles.

- *Acción*: una acción expresa una operación de cambio de estado del sistema. La acción puede ser *concreta*, es decir, directamente comprensible para ser realizable por el destinatario del plan o *abstracta*, es decir, una acción que resume un conjunto de acciones concretas y que para que sea realizable debe ser refinada detallando cuáles son las acciones concretas que la forman. Las acciones se dividen también en acciones externas, de control y de reparación. Las acciones *externas* son ajenas al responsable de la gestión o mantenimiento del sistema. Las *acciones de control* actúan sobre los órganos de control para modificar el estado del sistema pero sin cambiar su estructura. Las *acciones de reparación* modifican estructura y/o componentes del sistema.

- *Plan*: un plan es un conjunto de acciones. Durante el razonamiento se manejan planes con acciones tanto concretas como abstractas. El término *sub-plan* corresponde a un plan que resuelve sólo una parte del problema.

El problema de planificación puede definirse de la siguiente forma:

Definición 6.1: *Planificar* tiene como fin encontrar un conjunto de acciones que aplicadas sobre un sistema dinámico permiten que se alcance un determinado objetivo.

Dicho objetivo puede expresarse (1) en forma de estado o (2) en forma de acción abstracta. En general el contenido de un plan puede ser:

- *una acción simple*: un término que identifica una única acción, por ejemplo: `evacuar, vender`,

- *una acción parametrizada*: un término que indica una acción con argumentos que parametrizan la acción, por ejemplo: `suministrar(amoxicilina, 500mg), invertir(fondo-inversión, 2.500€)`,

- *un conjunto de acciones:* un conjunto de acciones en donde no importa el orden,

- *un conjunto ordenado de acciones:* un conjunto de acciones que tienen que realizarse en un determinado orden (total o parcial),

- *un conjunto de acciones con régimen de control complejo:* un conjunto de acciones que se realizan de acuerdo con un régimen de control que puede incluir bucles (con condiciones de terminación) y alternativas (bifurcaciones),

- *un conjunto de acciones con asignación temporal:* el plan incluye los instantes temporales de comienzo y terminación de cada acción.

Los problemas en donde se tiene como resultado un plan con la forma primera y segunda (una única acción) pueden resolverse como un problema de clasificación. El último caso (con asignación temporal) puede resolverse como un problema de

asignación en donde los recursos son los diferentes intervalos temporales. El resto de los casos requiere utilizar métodos propios de planificación.

El término planificación se utiliza en el lenguaje usual para denominar diferentes tipos de tareas que en ocasiones no son realmente problemas de planificación propiamente dicha. Por ejemplo, planificar las guardias de un servicio de urgencias en un hospital es realmente un problema de asignación; planificar una ruta turística y planificar el programa de festejos de una población son problemas de configuración. En particular, planificación tiene similitudes importantes con configuración tal como ya indica [Clancey, 85] y, en ocasiones, es posible analizar el mismo problema bajo ambas perspectivas siendo normalmente una de ellas más natural.

Por ejemplo, planificar una ruta turística tiene como fin encontrar una secuencia de visitas en lugares turísticos, con estancias en hoteles, uso de medios de transporte, etc. Este problema puede considerarse como configuración en donde se manejan determinados componentes (visitas, estancias, etc.) que se eligen mediante funciones (diversión, contenido histórico, descanso, etc.) y preferencias del viajero. También podría contemplarse como un problema de planificación en donde las visitas y estancias son acciones a realizar y se eligen en función del efecto que producen en el sistema donde se aplican (el viajero). Esta segunda consideración, aunque correcta formalmente, es menos adecuada dado que centra la atención en los efectos que produce viajero lo cual es no es muy natural.

Por otra parte, en un caso de control de un robot, es preferible considerar el problema como planificación dado que se tiene muy presente el efecto que se tiene sobre el sistema receptor de las acciones (el robot). Otro ejemplo que presenta cierta dualidad es planificar inversiones. En una inversión, vista como un problema de planificación, las acciones pueden ser: invertir en bolsa, invertir en fondo de pensiones, invertir en inmuebles, etc. La atención puede centrarse en el patrimonio del inversor que es donde actúa el plan, estableciendo como objetivos un determinado incremento de su

valor. Pero puede verse también como un problema de configuración con un conjunto de componentes (las opciones de inversión antes citadas) que tienen ciertas funciones expresadas en forma de expectativas de beneficio. En este caso, ambos enfoques son razonables.

De forma general, en estos casos se tienen dos sistemas: un sistema *A* y un sistema objeto *O* sobre el que actúa *A*. Por ejemplo *A* es la inversión, la ruta turística o un recorrido de un robot y *O* es, respectivamente, el patrimonio del inversor, el viajero o el robot. En el enfoque de configuración la atención se centra en *A* sin hacer explícito el sistema *O* y sus estados (a cambio se habla de las funciones que debe cumplir *A* y sus componentes). En planificación la atención se centra en el sistema *O* y sus estados que dirigen la selección de las acciones que son los componentes de *A*.

	planificación (de A sobre O)	**configuración (de A)**
objetivo	un estado meta del sistema O, o bien una acción abstracta sobre O	el sistema A debe ofrecer ciertas funciones
resultado	plan de acciones	estructura de componentes
elemento básico	acción sobre un sistema O	componente del sistema A
sistema principal	el sistema O	el sistema A
criterio de selección de un elemento	estados en el sistema O, acciones sobre O	función que cumple A

Figura 6.1: Comparación entre planificación y configuración.

La figura 6.1 muestra una comparación de los dos tipos de problemas. El resultado de planificar es un plan cuyos elementos son acciones sobre un determinado sistema *O*. En configuración el resultado es la estructura de *A* cuyos elementos son los componentes del sistema.

A su vez, dentro de los problemas de planificación se pueden incluir otro tipo de tareas más específicas en función de los tipos de acciones que se generan:

- *Controlar* un sistema dinámico (por ejemplo un conjunto de embalses en una cuenca hidrográfica). En este caso, se trata de encontrar un conjunto de acciones de control a partir de un estado inicial que presenta problemas descritos por un conjunto de síntomas y causas. El objetivo es la eliminación de dichos problemas.

- *Reparar* un sistema de componentes dinámicos. Se trata de generar un plan para eliminar las averías que presenta un sistema de componentes dinámicos. El estado inicial incluye los síntomas y las averías detectadas.

En este capítulo se presenta un método de planificación que se encuadra dentro de los denominados planificadores HTN (*Hierarchical Task Network*). En este tipo de planificadores el objetivo se define no como un estado a alcanzar sino como una tarea a realizar (una acción abstracta), por ejemplo "invertir 10.000 euros" o "viajar a Londres". El planificador cuenta con conocimiento sobre diferentes formas de realizar acciones abstractas que permite descomponerlas en acciones más sencillas.

La planificación HTN es una de las soluciones más utilizadas en aplicaciones prácticas complejas. Su planteamiento y caracterización ha sido realizado y extendido por diversos autores [Sacerdoti, 75; Tate, 77; Wilkins, 88; Erol et al., 94; Nau et al., 03]. La idea de planificación jerárquica ha sido también considerada en sistemas como MOLGEN [Friedland, 79] (*skeletal plan refinement*). Un enfoque centrado en problemas de configuración que sigue la misma idea de construcción jerárquica la utilizan Brown y Chandrasekaran [Brown, Chandrasekaran 89] en donde se plantean los conceptos de planes de diseño y especialistas.

Desde el punto de vista de representación, este método tiene las siguientes características:

- *Representación explícita de criterios de selección de las acciones.* Se representa explícitamente conocimiento sobre la forma de seleccionar las acciones que forman los planes.

- *Representación explícita del efecto de las acciones.* Se representa el efecto que tienen las acciones concretas sobre el sistema que actúa el planificador.

- *Manejo de estructuras abstractas de planes.* Se dispone de estructuras abstractas que se refinan progresivamente hasta alcanzar el plan final mediante un descenso jerárquico.

Para seleccionar acciones o grupos de acciones, los planificadores del campo de inteligencia artificial utilizan un modelo explícito con los criterios en los que se basa la selección. La selección de las acciones pueden ser de dos tipos:

- *hacia atrás (backward):* selección de las acciones por sus fines, es decir, por el efecto que producen en cuanto a la transformación del estado sobre el que actúan,

- *hacia delante (forward):* selección de las acciones por condiciones de aplicabilidad sobre el estado del sistema.

En el primer caso, la estrategia de búsqueda consiste en que, dado un estado objetivo, se eligen las acciones que permiten alcanzarlo. Esta es la estrategia tradicionalmente denominada análisis medios fines y utilizada por ejemplo en el método STRIPS y que, en problemas complejos, puede presentar problemas de eficiencia. En el otro caso se tiene un enfoque cuya eficiencia puede mejorarse significativamente si el

espacio de acciones candidatas en cada caso se reduce mediante acciones abstractas tal como plantea la planificación HTN que se describe en el presente capítulo, lo cual permite abordar eficazmente problemas de complejidad importante.

La figura 6.2 muestra un ejemplo que ilustra la forma de operar del método[1]. El método desarrolla un proceso de búsqueda, desarrollando un árbol Y-O (un árbol con ramas en conjunción y ramas en disyunción) como el de la figura en donde inicialmente, se plantea como objetivo realizar una acción abstracta (acción-0). Utilizando un determinado conocimiento, se selecciona una primera estrategia para realizar dicha acción. En el ejemplo, dicha estrategia se denomina estrategia-1 (por ejemplo, si consideramos el problema de viajar a una determinada ciudad como Londres, estrategia-1 puede ser viajar-en-avión). Este plan está formado por tres acciones: {acción-1, acción-2, acción-3}. Dichas acciones se deben realizar en el orden indicado.

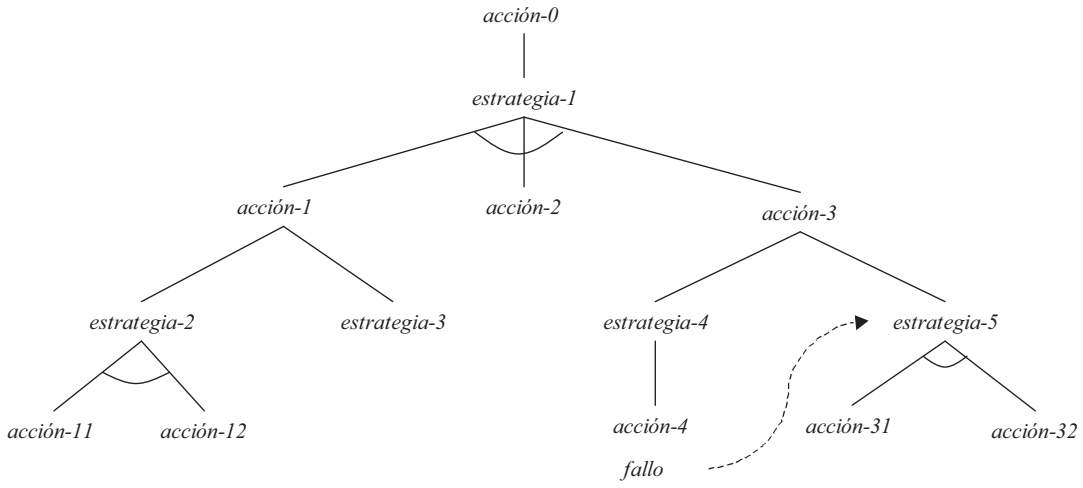


Figura 6.2: Ejemplo de árbol de búsqueda desarrollado por el método de planificación jerárquica HTN.

[1] En los planificadores HTN normalmente se utiliza el término de *tarea* para lo que en el presente texto se denomina *acción* y el término *método* para lo que en el texto se denomina *estrategia*. Se ha preferido aquí seguir esta denominación alternativa con el fin de no confundir con el significado que se utiliza de forma generalizada en el campo de ingeniería del conocimiento para *tarea* y *método*.

En el árbol se muestran en orden de izquierda a derecha y con un arco que une las ramas indicando con ello que están en conjunción. Las acciones acción-1 y acción-3 se consideran abstractas y la acción acción-2 se considera concreta (acción-1 podría ser ir-al-aeropuerto, acción-2 podría ser tomar-vuelo-a-Londres y acción-3 podría ser ir-al-hotel). Cada acción abstracta debe ser refinada repitiendo el proceso, buscando un plan más detallado. Por ejemplo, acción-1 puede llevarse a cabo con estrategia-2 (ir-al-aeropuerto-en-coche) o con estrategia-3 (ir-al-aeropuerto-en-metro). En el árbol, estas dos alternativas se indican con ramas que no están unidas por un arco, lo que quiere decir que se entiende de forma disyuntiva.

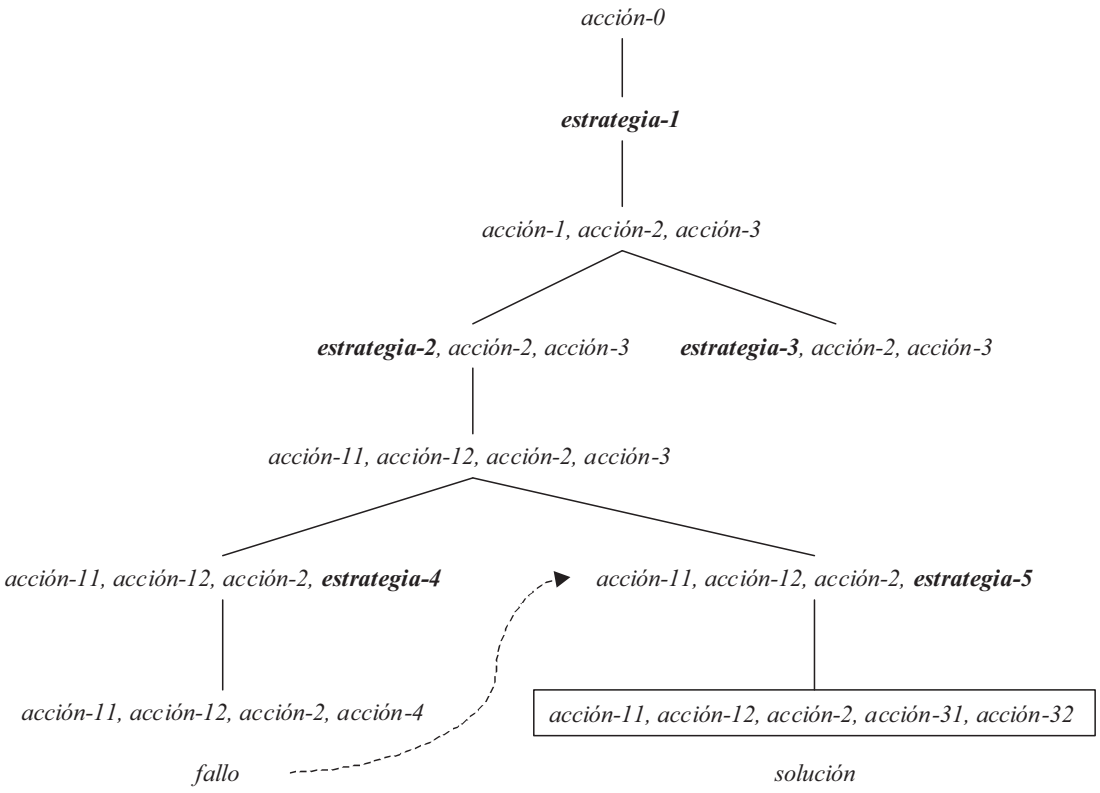


Figura 6.3: Árbol de búsqueda alternativo para el ejemplo de la figura 6.2.

El proceso refina la estrategia-2 y, dado que tiene éxito, no continúa con estrategia-3. A continuación, el proceso de búsqueda continúa con el refino de acción-3, obteniendo dos alternativas estrategia-4 y estrategia-5. Dado que la rama por estrategia-4 falla, la búsqueda retrocede para estudiar la rama alternativa correspondiente a estrategia-5. Finalmente, el conjunto de acciones correspondiente al plan final se encuentra en las hojas de dicho árbol: {acción-11, acción-12, acción-2, acción-31, acción-32}.

La figura 6.3 muestra una representación gráfica alternativa de la búsqueda que permite evitar a confusiones como la que muestra el siguiente caso. En el ejemplo de la figura 6.2, una vez encontrada la primera opción de refino de la acción abstracta acción-1 con las acciones concretas {acción-11, acción-12} se continúa con el refino de la acción abstracta acción-3. Pero, si por la rama de acción-3 no se encontrase solución, el gráfico puede sugerir erróneamente que no hay solución para la estrategia-1, dado que acción-3 está en conjunción con acción-1 y acción-2. Lo correcto es, sin embargo, volver a la siguiente opción de acción-1 (estrategia-3 en este caso) y continuar la búsqueda a partir de ahí. Esto es debido a que el fallo de acción-3 puede estar condicionado por el estado del sistema resultante de las decisiones tomadas para acción-1. Debido a ello, es más conveniente visualizar el proceso de búsqueda tal como muestra el árbol de la figura 6.3 en donde cada camino del árbol contiene una evolución del estado del sistema consistente con el plan desarrollado en dicho camino.

6.2 Organización del conocimiento

De acuerdo con la descripción anterior el conocimiento en este método se organiza en tres tipos: (1) estrategias, (2) acciones, (3) efectos. El conocimiento de estrategias incluye criterios de naturaleza heurística sobre selección de estrategias. Para expresar

este conocimiento es posible hacer uso de sentencias de tipo condicional, por ejemplo en forma de reglas, como:

<acción> y <condiciones> → *<estrategia>*

En dicha sentencia se indica en el antecedente la acción abstracta que se pretende realizar seguida de las condiciones que se deben dar. En el consecuente se indica la estrategia. Las condiciones incluyen afirmaciones condicionales sobre el estado del sistema sobre el que se aplica el plan. Por ejemplo:

```
ir(X) y
dinero(persona) > 25 y
posición(persona) = Y y distancia(X, Y) < 15
→ taxi(Y, X)
```

en donde la acción abstracta ir(X), que significa ir al lugar X, se puede realizar con la estrategia taxi(Y, X), que significa ir en taxi del lugar Y al lugar X, si se cumplen las condiciones de que el dinero disponible de la persona sea más de 25 euros y la distancia entre la posición actual de la persona y X sea menor de 15 Km. En general, para una acción puede haber más de una estrategia aplicable.

El conocimiento de acciones indica las acciones de las que consta cada estrategia. La representación puede realizarse indicando, para cada estrategia la lista ordenada de acciones (abstractas o concretas) que lo forman, de la forma siguiente:

<estrategia> → *{acción-1, acción-2, ..., acción-N}*

La representación puede incluir variables con el fin de dar más generalidad a las afirmaciones. Por ejemplo:

```
taxi(X, Y)
→ {llamar-taxi(X), ir-en-taxi(X,Y), pagar-taxi(X,Y)}
```

Las base de efectos indica los efectos que tiene en el sistema la realización de acciones concretas. Para expresar los efectos se pueden utilizar sentencias con el siguiente formato:

<acción> y <condiciones> → *<efecto>*

En dicha sentencia la acción corresponde al identificador (opcionalmente con variables) de una acción concreta. Las condiciones son afirmaciones condicionales sobre atributos de estado. El efecto se expresa mediante valores que se asignan a atributos de estado. Por ejemplo:

```
ir-en-taxi(X, Y) y posición(persona) = X y posición(taxi) = X →
posición(persona) = Y, posición(taxi) = Y

pagar-taxi(X, Y) y dinero(persona) = Z
→ dinero (persona) = (Z - 2 + 0.8 * distancia(X, Y))
```

Tipo de conocimiento		**Explicación**
Estrategias	*Significado*	Criterios de selección de estrategias.
	Representación	Sentencias en forma de reglas con el formato de <acción>, <condiciones> → <estrategia>

	Ejemplos	`ir(X) y`
		`dinero(persona) > 25 y`
		`posición(persona) = Y y distancia(X, Y) < 20`
		`→ taxi(Y, X)`

Acciones	*Significado*	Acciones de las que consta cada estrategia expresadas en forma de lista ordenada.
	Representación	Sentencias en forma de reglas con el formato de <estrategia> → {<acción-1>, ..., <acción-N>}

	Ejemplos	`taxi(X, Y)`
		`→ {llamar-taxi(X),`
		`ir-en-taxi(X,Y), pagar-taxi(X,Y)}`

Efectos	*Significado*	Efectos que tienen las acciones concretas.
	Representación	Sentencias en forma de reglas con el formato de <acción>, <condiciones> → <efecto>

	Ejemplos	`ir-en-taxi(X, Y) y`
		`posición(persona) = X y posición(taxi) = X`
		`→ posición(persona) = Y, posición(taxi) = Y`

Figura 6.4. Organización del conocimiento.

Nótese que algunas afirmaciones en los efectos pueden implicar la supresión de ciertos valores. Por ejemplo, la existencia de A1=b en los efectos (junto a A1=a en las condiciones) significa que si realiza esta acción, en la base de hechos previamente existirá A1=a pero, como consecuencia de la realización de la acción, el valor A1=a será sustituido por A1=b, lo cual supone no sólo que se añade cierta información sino también que se suprime otra. En general, mientras no se indique lo contrario, se asume que los valores de atributos son excluyentes.

Como caso particular de las sentencias sobre efectos se admiten sentencias de verificación de condiciones con efecto nulo como las siguientes:

```
<acción> y <condiciones>
```

El significado de esta sentencia es que, para que sea aplicable la acción concreta, se deben dar las condiciones expresadas.

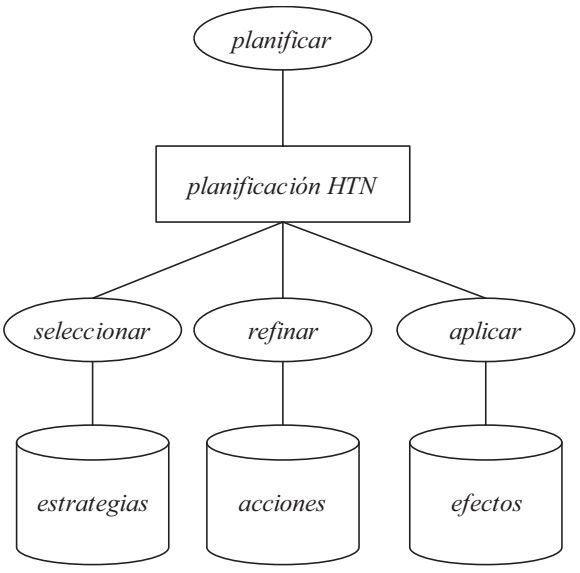


Figura 6.5: Estructura del método considerada en el algoritmo 1.

6.3 Inferencia

Para este método se van a considerar dos variantes de inferencia. La primera, la más compleja, se aplica a problemas de planificación. La segunda corresponde a una adaptación del método al problema de configuración.

6.3.1. Algoritmo 1: Planificación jerárquica HTN

Los pasos de inferencia que se consideran en este algoritmo son tres: seleccionar, refinar y aplicar (figuras 6.5 y 6.6). El paso de inferencia *seleccionar* tiene como fin determinar una o varias estrategias candidatas a partir de la acción abstracta que se pretende realizar y el estado del sistema. Recibe como entrada la acción abstracta y los hechos que representan el estado actual del sistema. Como resultado genera el conjunto de identificadores de estrategias candidatas. Por ejemplo, el paso de inferencia puede recibir como entrada la acción denominada acción-1(a) y el estado {A=a, B=b} y generar como resultado las estrategias posibles {estrategia-1(a), estrategia-2(b)}.

```
INFERENCIA seleccionar
       DATOS:                    acción, estado
       BASES DE CONOCIMIENTO:    estrategias
       RESULTADOS:               estrategias

INFERENCIA refinar
       DATOS:                    estrategia
       BASES DE CONOCIMIENTO:    acciones
       RESULTADOS:               subplanes

INFERENCIA aplicar
       DATOS:                    acción, estado
       BASES DE CONOCIMIENTO:    efectos
       RESULTADOS:               estado
```

Figura 6.6: Pasos de inferencia considerados en el algoritmo 1.

Dado que la sentencias de la base de conocimiento de estrategias normalmente incluyen variables, el proceso general de selección de una determinada estrategia para una determinada acción abstracta puede llevar a cabo una instanciación (particularización) de una estrategia general realizando una sustitución de las variables por constantes de acuerdo con el estado del sistema y de la acción a realizar.

Así por ejemplo, dada la acción objetivo ir(Atocha) (ir a la estación de Atocha), el estado {posición(persona)=Chamartín, dinero(persona)=40} y la sentencia de la base de estrategias:

```
ir(X) y
dinero(persona) > 25 y
posición(persona) = Y y distancia(X, Y) < 20
→ taxi(Y, X)
```

el paso de inferencia seleccionar debe realizar la sustitución <X/Atocha, Y/Chamartín> para obtener la instancia de estrategia taxi(Chamartín, Atocha).

El paso de inferencia *refinar* tiene como fin determinar cuál es el conjunto de acciones de una determinada estrategia. Para ello utiliza el conocimiento de acciones dando como resultado un conjunto de subplanes expresado como una lista de listas ordenadas de acciones. También en este caso es posible realizar un proceso de unificación que dé lugar a una sustitución de variables por constantes para obtener una instancia de subplan. Por ejemplo, se puede recibir como dato el identificador de estrategia estrategia-3(a) y se genera como subplanes {{acción-1(b)}, {E1(a), accion-1,E2(b),accion-2(c)}}.

El paso de inferencia *aplicar* analiza el efecto que tiene una determinada acción concreta. Recibe como entrada un estado, una acción concreta y, utilizando el conocimiento de efectos, genera como resultado un nuevo estado en donde pueden presentarse nuevos atributos con valores y/o algunos atributos pueden cambiar su

estado. Por ejemplo, dado el estado {posición(persona)=Atocha, dinero(persona)=40} y la acción concreta pagar-taxi(Chamartín, Atocha) el estado resultante puede ser {posición(persona)=Atocha, dinero(persona)=29} en donde se ha modificado el atributo dinero(persona) como resultado de dicha acción.

Rol dinámico	**Significado**	**Representación**	**Ejemplo**
`estado`	afirmaciones sobre el estado del sistema en forma de hechos que son dato o que han sido deducidos	conjunto de pares atributo-valor	`{A=a, B=b, C=d, D=d}`
`acción`	identificador de una acción; puede ser abstracta o concreta	estructura	`acción-A(a,b)`
`estrategia`	identificador de una estrategia	estructura	`estrategia-5(a,b)`
`plan`	lista ordenada de acciones concretas; se trata del plan que se va construyendo progresivamente hasta encontrar el plan final	conjunto ordenado estructuras	`{acción-A3(a,b), acción-P(c), acción-Q(d,e), acción-T}`
`guía`	lista ordenada de acciones (abstractas y concretas); sirve como guía al proceso de planificación	conjunto ordenado estructuras	`{acción-11(a), acción-P(c), acción-12(d,e)}`
`subplan`	lista ordenada de acciones (abstractas y concretas); corresponde a las acciones de las que consta una estrategia	conjunto ordenado estructuras	`{acción-11(a), acción-P(c), acción-12(d,e)}`
`éxito`	indica si ha tenido éxito la búsqueda del plan	valor de `{VERDADERO, FALSO}`	`VERDADERO`

Figura 6.7 : Roles dinámicos que intervienen en el proceso de inferencia.

La figura 6.7 resume el conjunto de roles dinámicos que manejan los pasos de inferencia. La figura 6.8 muestra en pseudo-código el ejemplo de algoritmo considerado. Básicamente este algoritmo, a partir de los datos del problema para cada acción, determina una estrategia haciendo razonamiento clasificativo. En general puede haber varias opciones de estrategias o incluso puede no haber ninguna opción, lo cual supone un rechazo. Dada una estrategia, se refina obteniendo los

subproblemas a resolver. Dichos subproblemas están formados por una lista ordenada de varias acciones. Cada acción plantea un subproblema que debe ser tratado de nuevo con alguna estrategia. Las acciones concretas son directamente realizables y se aplican para considerar su efecto en el estado del sistema.

```
METODO planificación-jerárquica-HTN
  DATOS: estado, acción
  RESULTADOS: estado, plan, éxito

ALGORITMO
1. guía = {acción}
2. planificar(estado, guía, {} -> estado, plan, éxito)

PROCEDIMIENTO planificar
  DATOS: estado, guía, plan
  RESULTADOS: estado, plan, éxito
1. IF guía = ϕ THEN éxito = VERDADERO
2. ELSE
3.   GET-FIRST(acción, guía)
4.   IF TIPO(acción) = concreta THEN
5.     aplicar(estado, acción -> estado)
6.     planificar(estado, guía, plan U {acción} -> estado, plan, éxito)
7.   ELSE
8.     seleccionar(acción, estado -> estrategias)
9.     éxito = FALSO
10.    WHILE (estrategias no ϕ) AND no(éxito)
11.      GET (estrategia, estrategias)
12.      refinar(estrategia -> subplanes)
13.      WHILE (subplanes no ϕ) AND no(éxito)
14.        GET (subplan, subplanes)
15.        planificar(estado, subplan U guía, plan -> estado, plan, éxito)
```

Figura 6.8: Ejemplo de algoritmo del método de planificación jerárquica HTN.

Cada vez que se realiza el paso *aplicar*, se anotan los efectos de una determinada acción concreta, modificando valores correspondientes al estado del sistema. Esto, en general, permitirá influir en la selección del resto de estrategias conforme continúa el proceso de búsqueda. Así, en el proceso de razonamiento, el estado del sistema va cambiando progresivamente conforme a la elección de las diferentes estrategias. No obstante, cuando se produce un fallo en un camino de búsqueda, la vuelta atrás para

desarrollar otro camino de búsqueda debe retraer los efectos correspondientes al primer camino. Esta retracción de efectos se lleva a cabo en el proceso general tal como se muestra en el algoritmo del método mediante la programación de un esquema recursivo.

El algoritmo tiene un cuerpo principal que llama a un procedimiento. Este procedimiento, denominado *planificar*, realiza un paso de resolución del problema de planificación y es recursivo de forma que, al llamarse a sí mismo con nuevos datos va completando el proceso de búsqueda. Los argumentos de entrada correspondientes el estado del sistema y el plan en curso son también argumentos de salida para ir recogiendo los cambios que se van produciendo en ambos. Como entrada se tiene también la variable que guía el proceso de planificación indicando las acciones (concretas o abstractas) que están pendientes de analizar. Además, se tiene otra variable de salida (éxito) con valor VERDADERO o FALSO para indicar si se ha encontrado solución. En la variable *plan* se almacena en cada momento el plan en curso formado en general por acciones concretas.

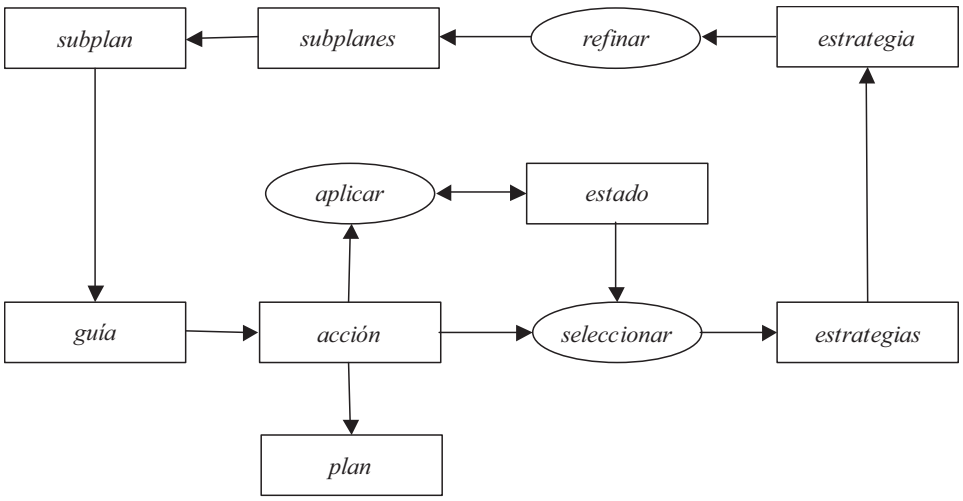


Figura 6.9: Estructura de inferencia del ejemplo de algoritmo 1.

Aspectos de búsqueda y eficiencia del método

El algoritmo que se presenta aquí corresponde a una versión de la planificación HTN a la que se denomina TFD (*Total-order Forward Decomposition*) en donde cada conjunto de acciones (abstractas o concretas) que sirve como guía al proceso de planificación es un conjunto con *orden total* que se procesa *hacia delante*. El algoritmo puede generalizarse para tratar conjuntos con orden parcial en donde cada estrategia propone subplanes que no imponen un orden total sino parcial. Esta segunda versión se denomina PFD (*Partial-order Forward Decomposition*). Existen además otras versiones más generales de planificación HTN que no imponen la descomposición hacia delante, manejando redes con restricciones entre las acciones. Para una visión general de métodos de planificación puede consultarse el libro [Ghallab et al., 04].

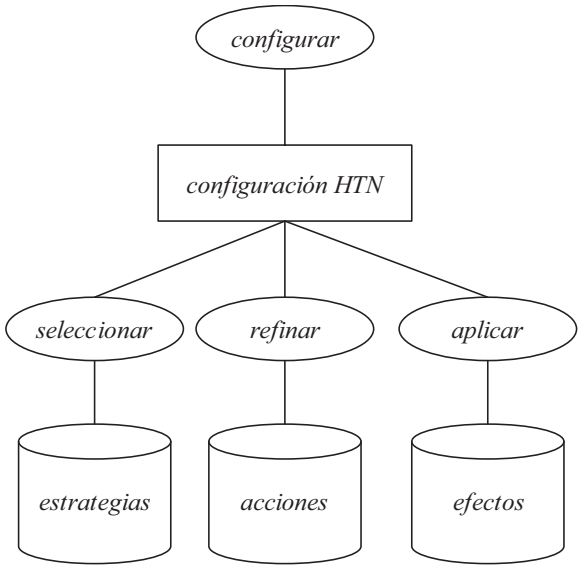


Figura 6.10: Estructura del método considerada en el algoritmo 2.

Debido a su eficiencia, los planificadores HTN han dado lugar a un número importante de aplicaciones prácticas [Wilkins, desJardins, 01]. Por ejemplo, aplicaciones tales como planificación de líneas de producción, gestión de crisis y logística, planificación en naves espaciales, planificación de procesos de fabricación,

planes de evacuación, el juego de bridge y planificación en robótica. Junto a aplicaciones prácticas se han desarrollado también herramientas software generales tales como Nonlin, SIPE-2, O-Plan, UMCP y SHOP2[2]. Entre ellos, SHOP2 es una de las herramientas eficientes para resolver problemas de planificación que más aceptación han tenido. Su popularidad aumentó debido a que fue una de las herramientas premiadas en la Competición Internacional de Planificación de 2002.

La planificación HTN comparada con la planificación clásica (por ejemplo, de tipo STRIPS) puede resolver problemas de planificación varios órdenes de magnitud más rápido. En general, los estudios sobre expresividad, completitud y complejidad computacional de la planificación HTN muestran ventajas frente a planificación clásica [Ghallab et al., 04]. Por ejemplo, los planificadores HTN utilizan un lenguaje más expresivo (del tipo gramática de contexto libre) que los que presentan los planificadores clásicos (gramática regular). Además, un método clásico como STRIPS no es completo dado que existen problemas de planificación que, a diferencia de los planificadores HTN, no puede resolver (por ejemplo, el problema de intercambio de registros). Como contrapartida, el desarrollador en un planificador HTN debe construir bases de conocimiento que incluyan no sólo los operadores (con las condiciones y efectos) como en la planificación clásica sino también las estrategias posibles que reducen los espacios de búsqueda de dichos operadores.

Por otra parte, en la versión de método que aquí se describe, el paso de inferencia *seleccionar* utiliza una representación de reglas para la elección de estrategias de cada acción. Los diferentes tipos de acciones pueden contemplarse organizados en niveles de abstracción con un enfoque jerárquico. Para cada acción (es decir, para cada nodo del árbol) se tiene un conjunto de reglas que se utiliza para seleccionar la mejor estrategia a ese nivel de abstracción. Es interesante hacer notar que la selección de estrategia para cada acción es la resolución de un problema de tipo

[2] Se pueden obtener copias de la mayor parte de dichas herramientas en sus correspondientes páginas web. Por ejemplo, la página de SHOP2 es `www.cs.umd.edu/projects/shop`.

clasificativo. Aunque aquí se utiliza una representación en reglas con un determinado formato, podría generalizarse para contemplar otro tipo de organización del conocimiento para resolver el problema de clasificación (incluyendo, por ejemplo, fases de abstracción y asociación como se realiza en clasificación heurística).

```
INFERENCIA seleccionar
       DATOS:                    acción, diseño
       BASES DE CONOCIMIENTO:    estrategias
       RESULTADOS:               estrategias

INFERENCIA refinar
       DATOS:                    estrategia
       BASES DE CONOCIMIENTO:    acciones
       RESULTADOS:               subplanes

INFERENCIA aplicar
       DATOS:                    acción, diseño
       BASES DE CONOCIMIENTO:    efectos
       RESULTADOS:               diseño
```

Figura 6.11: Pasos de inferencia considerados en el algoritmo 2.

6.3.2. Algoritmo 2: Configuración jerárquica HTN

El método de planificación jerárquica HTN puede adaptarse para ser aplicable en problemas de configuración en vez de problemas de planificación. Este apartado muestra una versión de dicho algoritmo para problemas de configuración.

Para aplicar el método de planificación jerárquica HTN a un problema de configuración se considera que la configuración debe realizarse mediante ciertas acciones de diseño. La organización de dichas acciones da lugar lo que se denomina *planes de diseño* y permite una descomposición del problema de configuración global del sistema en subproblemas de configuración de partes del sistema. La idea de planes de diseño la aplicaron Brown y Chandrasekaran [Brown, Chandrasekaran, 89] introduciendo el concepto de especialista en el lenguaje para diseño DSPL. El

concepto de planes de diseño se encuentra también en otras propuestas [Friedland, 79; Rich, 81; Johnson, Soloway, 85; Mittal et. al, 86]. La idea de aplicar planificación HTN en configuración, por ejemplo, ha sido considerada en el problema de configuración de equipos [Agosta, 95].

Rol dinámico	**Significado**	**Representación**	**Ejemplo**
`diseño`	afirmaciones sobre el diseño en curso, inicialmente contiene sólo las especificaciones	conjunto de pares atributo-valor	`{A=a, B=b, C=d, D=d}`
`especificaciones`	especificaciones del diseño a realizar correspondientes a funciones y preferencias	conjunto de pares atributo-valor	`{A=a, B=b}`
`estrategia`	identificador de una estrategia	estructura	`estrategia-5(a,b)`
`acción`	identificador de una acción de diseño; puede ser abstracta o concreta	estructura	`acción-A(a,b)`
`guía`	lista ordenada de acciones de diseño (abstractas y concretas); sirve como guía al proceso de planificación	conjunto ordenado estructuras	`{acción-11(a), acción-P(c), acción-12(d,e)}`
`subplan`	lista ordenada de acciones de diseño (abstractas y concretas); corresponde a las acciones de las que consta una estrategia de diseño	conjunto ordenado estructuras	`{acción-11(a), acción-P(c), acción-12(d,e)}`
`éxito`	indica si ha tenido éxito la búsqueda del plan	valor de `{VERDADERO, FALSO}`	`VERDADERO`

Figura 6.12: Roles dinámicos que intervienen en el proceso de inferencia.

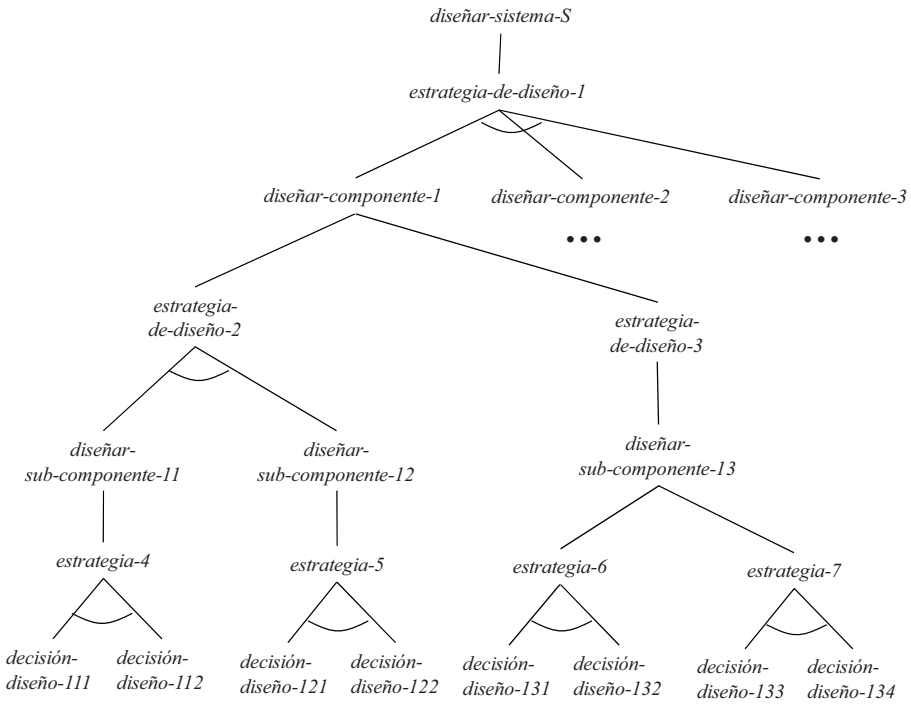


Figura 6.13: Organización del esquema de configuración con planes de diseño.

La forma en que se aplica el modelo planificación general en problemas de configuración es la siguiente (figura 6.13):

- Se utilizan acciones de diseño para distribuir el conocimiento de configuración de acuerdo con los diferentes componentes del sistema a diseñar. Cada acción es responsable de cómo diseñar una parte del sistema (un componente). Los acciones de niveles de abstracción más altos tratan aspectos más generales de los componentes y los de niveles más bajos con componentes más específicos. La acción raíz es responsable del diseño del sistema completo.

- Las acciones concretas fijan decisiones parciales (y tentativas) de diseño que quedan recogidas en los efectos de dichas acciones.

- Los atributos de estado se utilizan para recoger los requisitos funcionales, preferencias y las características del diseño del sistema (componentes y estructura).

- Las sentencias de selección de estrategias incluyen en las condiciones expresiones sobre requisitos funcionales, preferencias o estados de diseño. Los planes de las estrategias son *planes de diseño* que indican cómo las acciones de diseño de alto nivel delegan en acciones de diseño de niveles inferiores.

```
METODO configuración-jerárquica-HTN
  DATOS: especificaciones
  RESULTADOS: diseño, éxito

ALGORITMO
1. guía = {acción más general que representa el diseño total de sistema}
2. configurar(especificaciones, guía -> diseño, éxito)

PROCEDIMIENTO configurar
  DATOS: diseño, guía
  RESULTADOS: diseño, éxito
1.  IF guía = ϕ THEN éxito = VERDADERO
2.  ELSE
3.    GET-FIRST(acción, guía)
4.    IF TIPO(acción) = concreta THEN
5.      aplicar(diseño, acción -> diseño)
6.      configurar(diseño, guía -> diseño, éxito)
7.    ELSE
8.      seleccionar(acción, diseño -> estrategias)
9.      éxito = FALSO
10.     WHILE (estrategias no ϕ) AND no(éxito)
11.       GET (estrategia, estrategias)
12        refinar(estrategia -> subplanes)
13.       WHILE (subplanes no ϕ) AND no(éxito)
14.         GET (subplan, subplanes)
15.         configurar(diseño, subplan U guía -> diseño, éxito)
```

Figura 6.14: Ejemplo de algoritmo del método de configuración jerárquica HTN.

Los pasos de inferencia que se consideran en este método son también tres (seleccionar, refinar y aplicar) (figuras 6.10 y 6.11) y tienen un significado muy similar al caso anterior, con la correspondiente adaptación al caso de configuración. El paso de inferencia *seleccionar* tiene como fin obtener soluciones de diseño. Recibe como entrada los hechos que representan el estado actual del diseño y la acción de diseño. Tras realizar la búsqueda en la base de conocimiento de estrategias, genera como resultado un conjunto estrategias posibles que indican formas de delegar en otras acciones de diseño de niveles inferiores.

El paso de inferencia *refinar* coincide con el paso de inferencia de la versión anterior del algoritmo, excepto en el detalle de que la lista que genera se entiende como lista de acciones de diseño. El paso de inferencia *aplicar* también es similar al caso de planificación excepto que el efecto es sobre el diseño del sistema. La figura 6.12 resume el conjunto de roles dinámicos que manejan los pasos de inferencia.

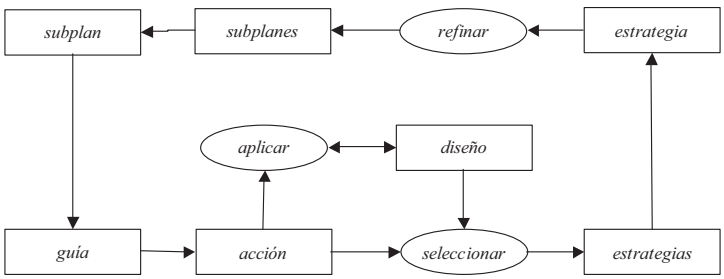


Figura 6.15: Estructura de inferencia del ejemplo de algoritmo 2.

En la figura 6.15. se muestra en pseudo-código el ejemplo de algoritmo planteado. El algoritmo opera de una forma similar a la versión anterior excepto que en este caso, en vez de manejar un plan que se va detallando progresivamente, se maneja un plan guía que va indicando la forma en que se van delegando las decisiones de diseño. Dicho plan guía va creciendo progresivamente, pero a medida que se completan decisiones de diseño se va reduciendo hasta que su contenido es vacío cuando se ha encontrado un diseño válido. Se maneja un cuerpo de algoritmo que llama un procedimiento recursivo denominado *configurar*.

6.4 Ejemplos

6.4.1. Ejemplo 1: Ejemplo de aproximación

Considérese el siguiente ejemplo que tiene como base de conocimiento de estrategias las siguientes sentencias:

```
acción-0 y A1 = a y A2 = a → estrategia-1(a)
acción-0 y A1 = a y A2 = b → estrategia-1(b)
acción-0 y A1 = b y A2 = X → estrategia-2(X)

acción-1(X) y A3 = b y A4 = b → estrategia-3(X, a)
acción-1(X) y A4 = b          → estrategia-4(X, c)
acción-1(X) y A3 = a y A4 = a → estrategia-3(a, a)

acción-11(P,X) y A3 = a y A4 = b → estrategia-4(X,a)
acción-11(X,Q) y A3 = a y A5 = a → estrategia-5(X)
acción-11(P,X) y A3 = b y A5 = a → estrategia-5(a)

acción-12(X)          → estrategia-5(X)
acción-12(X) y A3 = a → estrategia-4(a,X)
acción-12(X) y A4 = a → estrategia-4(X,a)

acción-2(X) y A2 = a y A7 = Y → estrategia-5(a,Y)
acción-2(X) y A2 = b y A7 = Y → estrategia-6(X,Y)
acción-2(X) y A2 = c          → estrategia-6(X,b)
```

La base de conocimiento de acciones es la siguiente:

```
estrategia-1(X)
→ {acción-A3(X), acción-1(X), acción-P(X), acción-2(X)}

estrategia-2(X)
→ {acción-A3(c), acción-Q(X), acción-2(X)}

estrategia-3(X,Y)
→ {acción-A5(b), acción-11(X,Y), acción-Q(Y)}

estrategia-4(X,Y)
→ {acción-A5(c), acción-12(X), acción-Q(Y)}

estrategia-5(X)
→ {acción-A6(b), acción-R, acción-S(X)}

estrategia-6(X, Y)
→ {acción-T(Y), acción-U(X, Y)}
```

La base de conocimiento de efectos es:

```
acción-A3(X)           → A3 = X
acción-A5(X)           → A5 = X
acción-A6(X) y A4 = Y → A4 = X,  A6 = Y
```

Para un determinado problema se cuenta con los siguientes hechos iniciales correspondientes al estado del mundo:

```
{A1=a, A2=b, A4=b, A7=a}
```

El proceso de resolución del problema es el siguiente (se muestra el orden de llamadas a los pasos de inferencia indicando los cambios que se producen en los conjuntos que maneja el algoritmo):

```
estado = {A1=a, A2=b, A4=b, A7=a}
guía = {acción-0}
acción = acción-0
1. seleccionar(acción, estado -> estrategias)
   estrategias = {estrategia-1(b)}
   estrategia = estrategia-1(b)
2. refinar(estrategia -> subplanes)
   subplanes = {{acción-A3(b), acción-1(b),
                 acción-P(b), acción-2(b)}}
   subplan = {acción-A3(b), acción-1(b), acción-P(b), acción-2(b)}
   guía = {acción-A3(b), acción-1(b), acción-P(b), acción-2(b)}
   acción = acción-A3(b)
3. aplicar(estado, acción -> estado)
   estado = {A1=a, A2=b, A3=b, A4=b, A7=a}
   plan = {acción-A3(b)}
   guía = {acción-1(b), acción-P(b), acción-2(b)}
```

```
   acción = acción-1(b)
4. seleccionar(acción, estado -> estrategias)
   estrategias = {estrategia-3(b,a), estrategia-4(b,c)}
   estrategia = estrategia-3(b,a)
5. refinar(estrategia -> subplanes)
   subplanes = {{acción-A5(b), acción-11(b,a), acción-Q(a)}}
   subplan = {acción-A5(b), acción-11(b,a), acción-Q(a)}
   guía = {acción-A5(b), acción-11(b,a), acción-Q(a),
           acción-P(b), acción-2(b)}
   acción = acción-A5(b)
6. aplicar(estado, acción -> estado)
   estado = {A1=a, A2=b, A3=b, A4=b, A5=b, A7=a}
   plan = {acción-A3(b), acción-A5(b)}
   guía = {acción-11(b,a), acción-Q(a), acción-P(b), acción-2(b)}
   acción = acción-11(b,a)
7. seleccionar(acción, estado -> estrategias)
   estrategias = {}
   se produce un FALLO por esta rama
   se vuelve a la opción alternativa en paso 4
   estado = {A1=a, A2=b, A3=b, A4=b, A7=a}
   plan = {acción-A3(b)}
   estrategia = estrategia-4(b,c)
8. refinar(estrategia -> subplanes)
   subplanes = {{acción-A5(c), acción-12(b), acción-Q(c)}}
   subplan = {acción-A5(c), acción-12(b), acción-Q(c)}
   guía = {acción-A5(c), acción-12(b), acción-Q(c),
           acción-P(b), acción-2(b)}
   acción = acción-A5(c)
```

```
9. aplicar(estado, acción -> estado)
   estado = {A1=a, A2=b, A3=b, A4=b, A5=c, A7=a}
   plan = {acción-A3(b), acción-A5(c)}
   guía = {acción-12(b), acción-Q(c), acción-P(b), acción-2(b)}
   acción = acción-12(b)
10.seleccionar(acción, estado -> estrategias)
   estrategias = {estrategia-5(b)}
   estrategia = estrategia-5(b)
11.refinar(estrategia -> subplan)
   subplan = {acción-A6(b), acción-R, acción-S(b)}
   guía = {acción-A6(b), acción-R, acción-S(b),
           acción-Q(c), acción-P(b), acción-2(b)}
   acción = acción-A6(b)
12.aplicar(estado, acción -> estado)
   estado = {A1=a, A2=b, A3=b, A4=b, A5=c, A6=b, A7=a}
   plan = {acción-A3(b), acción-A5(c), acción-A6(b)}
   guía = {acción-R, acción-S(b),acción-Q(c),
           acción-P(b), acción-2(b)}
   acción = acción-R
13.aplicar(estado, acción -> estado)
   plan = {acción-A3(b), acción-A5(c), acción-A6(b),acción-R}
   guía = {acción-S(b),acción-Q(c), acción-P(b), acción-2(b)}
   acción = acción-S(b)
14.aplicar(estado, acción -> estado)
   plan = {acción-A3(b), acción-A5(c),
           acción-A6(b),acción-R, acción-S(b)}
   guía = {acción-Q(c), acción-P(b), acción-2(b)}
   acción = acción-Q(c)
```

```
15.aplicar(estado, acción -> estado)
   plan = {acción-A3(b), acción-A5(c),
           acción-A6(b),acción-R, acción-S(b),acción-Q(c)}
   guía = {acción-P(b), acción-2(b)}
   acción = acción-P(b)
16.aplicar(estado, acción -> estado)
   plan = {acción-A3(b), acción-A5(c), acción-A6(b),
           acción-R, acción-S(b),acción-Q(c),acción-P(b)}
   guía = {acción-2(b)}
   acción = acción-2(b)
17.seleccionar(acción, estado -> estrategias)
   estrategias = {estrategia-6(b,a)}
   estrategia = estrategia-6(b,a)
18.refinar(estrategia -> subplanes)
   subplanes = {{acción-T(a),acción-U(b,a)}}
   subplan = {acción-T(a),acción-U(b,a)}
   guía = {acción-T(a),acción-U(b,a)}
   acción = acción-T(a)
19.aplicar(estado, acción -> estado)
   plan = {acción-A3(b), acción-A5(c), acción-A6(b),acción-R,
           acción-S(b),acción-Q(c),acción-P(b),acción-T(a)}
   guía = {acción-U(b,a)}
   acción = acción-U(b,a)
20.aplicar(estado, acción -> estado)
   plan = {acción-A3(b), acción-A5(c), acción-A6(b),acción-R,
           acción-S(b),acción-Q(c),acción-P(b),acción-T(a),
           acción-U(b,a)}
   guía = {}
   éxito = VERDADERO
   FIN
```

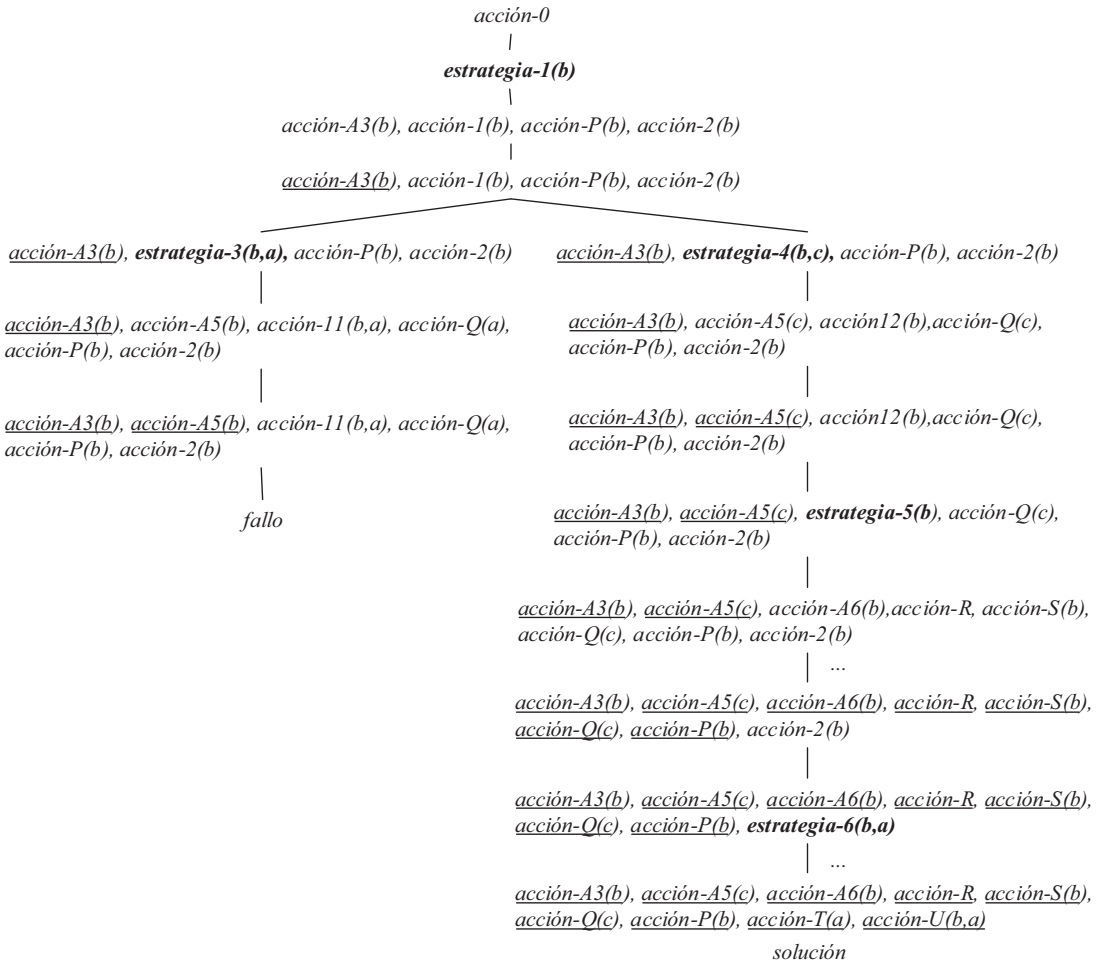


Figura 6.16: Árbol de búsqueda desarrollado en el ejemplo 1.

La figura 6.16 muestra el árbol de búsqueda desarrollado en el proceso de resolución del problema. Como resultado, se genera el siguiente plan:

```
plan = {acción-A3(b), acción-A5(c), acción-A6(b),acción-R,
        acción-S(b),acción-Q(c),acción-P(b),acción-T(a),
        acción-U(b,a)}
```

6.4.2. Ejemplo 2: Diseño de cilindros de aire comprimido

El sistema AIR-CYL resuelve un problema de configuración en el dominio de diseño de cilindros de aire comprimido. Este sistema fue propuesto por Brown y Chandrasekaran [Brown, Chandrasekaran, 89] como aplicación a dominio específico de un planteamiento general para problemas de diseño. Dicho planteamiento se definió con el lenguaje DSPL (*Design Specialists and Plans Language*) con el que se construyó AIR-CYL.

El problema que resuelve AIR-CYL puede considerarse como un caso de configuración que presenta similitudes con el que se resuelve por el método de configuración jerárquica HTN. Al igual que dicho método, se manejan planes de diseño que se completan y extienden progresivamente en un descenso jerárquico. Cada tipo de acción abstracta la determina en la terminología de AIR-CYL un *especialista*. Un especialista es un módulo que encapsula conocimiento especializado en seleccionar un tipo de acción de diseño. Hay un especialista para cada tipo de acción y, a su vez, un tipo de acción para cada decisión de diseño a realizar.

La figura 6.17 muestra un ejemplo entradas y salidas proporcionadas por AIR-CYL. AIR-CYL como otros sistemas para diseño utiliza como entrada un conjunto de datos que describen los requisitos del sistema a diseñar. En este caso, por ejemplo, se indican las características del lugar en donde se instalará el cilindro (dimensiones del espacio, máxima temperatura, entorno corrosivo, etc.), requisitos de montaje, valores límite de presión, etc. Para algunos valores se expresan tolerancias en forma de (LNGTH x y z). Como resultado del proceso, el sistema genera los valores específicos que describen cada uno de los componentes a diseñar como el resorte, el pistón, el tubo, etc.

```
REQUIREMENTS:

EnvelopeLength         ---- 7.83
EnvelopeHeight         ---- 1.5
EnvelopeWidth          ---- 1.75
MaxTemperature         ---- 250
OperatingMedium        ---- Air
OperatingPressureMax   ---- 60
OperatingPressureMin   ---- 30
RodLoad                ---- 1.4
Stroke                 ---- 1.75
RodThreadType          ---- UNF24
RodThreadLength        ---- 1.031
RodDiameter            ---- (LNGTH 0.312 0.0 2.e-3)
Environment            ---- Corrosive
Quality                ---- Reliable
MTBF                   ---- 100000
AirInletDiameter       ---- 0.374
MountingScrewSize      ---- (LNGTH 0.19 5.e-3 5.e-3)
MountingHoleToHole     ---- (LNGTH 0.625 5.e-3 5.e-3)
MaxFaceToMountingHoles ---- (LNGTH 0.31 5.e-3 5.e-3)

OUTPUTS:

SpringMaterial          ---- NIL
SpringOD                ---- 0.985
SpringID                ---- 0.77
SpringWireDiameter      ---- 0.215
SpringFreeLength        ---- NIL
SpringCompressedLength  ---- NIL
SpringInstalledLength   ---- NIL
SpringLoad              ---- NIL
SpringNumberOfCoils     ---- 11
SpringDeflectionPerCoil ---- NIL

HeadWidth                   ---- 1.5
HeadDepth                   ---- 0.97
HeadHeight                  ---- 1.5
HeadMaterial                ---- StainlessSteel
HeadScrewSize               ---- (LNGTH 0.19 5.e-3 5.e-3)
HeadCenterCenterDistance    ---- (LNGTH 0.625 5.e-3 5.e-3)
HeadMountingHoleDiameter    ---- (LNGTH 0.206 3.e-3 0.0)
HeadCounterSinkDiameter     ---- (LNGTH 0.37 1.e-2 1.e-2)
HeadMaxHtoFDistance         ---- (LNGTH 0.31 5.e-3 5.e-3)
HeadMountingHolesToFaceDistance ---- (LNGTH 0.2455 2.5e-3 2.5e-3)
...

PistonDiameter              ---- (LNGTH 1.212 4.e-3 0.0)
PistonMaterial              ---- Brass
PistonThickness             ---- 0.34375d
PistonRodHole               ---- 0.25d
PistonSpringSeatDepth       ---- 3.9e-2
PistonSpringSeatID          ---- 0.754375
PistonSpringSeatOD          ---- 1.00062
PistonSealType              ---- UCup
PistonSealSeatDiameter      ---- (LNGTH 0.885 0.0 1.e-3 ThreeDP)
PistonSealSeatWidth         ---- (LNGTH 0.156 1.e-2 1.e-2 ThreeDP)
...

RodDiameter           ---- (LNGTH 0.312 0.0 2.e-3)
RodLength             ---- 4.095
RodThreadLength       ---- 1.031
RodThreadType         ---- UNF24
RodMaterial           ---- StainlessSteel
RodPistonSeatDiameter ---- 0.247
RodPistonSeatLength   ---- 0.31
RodEndOfRodToHead     ---- 2.781

CapMaterial           ---- StainlessSteel
CapHeight             ---- 1.5
CapWidth              ---- 1.5
CapDepth              ---- 0.625
...

TubeMaterial      ---- StainlessSteel
TubeLength        ---- 3.5
TubeID            ---- 1.214
TubeOD            ---- 1.344

BumperMaterial        ---- StainlessSteel
BumperID              ---- 0.390625d
BumperOD              ---- 0.69
BumperFlangeDiameter  ---- 1.059
BumperFlangeThickness ---- 6.25e-2
```

Figura 6.17: Ejemplo parcial de entradas y salidas del sistema AIRCYL.

El lenguaje DSPL constituye una teoría de diseño basada en los conceptos de especialista y plan de diseño. Se utiliza la organización en *especialistas de diseño* para distribuir el conocimiento de configuración de acuerdo con los diferentes componentes del sistema a diseñar. Cada especialista incluye conocimiento sobre cómo diseñar una parte del sistema. Un especialista toma decisiones de diseño en la parte del sistema en el que está especializado. Sus decisiones pueden estar condicionadas por un contexto global generado por las decisiones de otros especialistas.

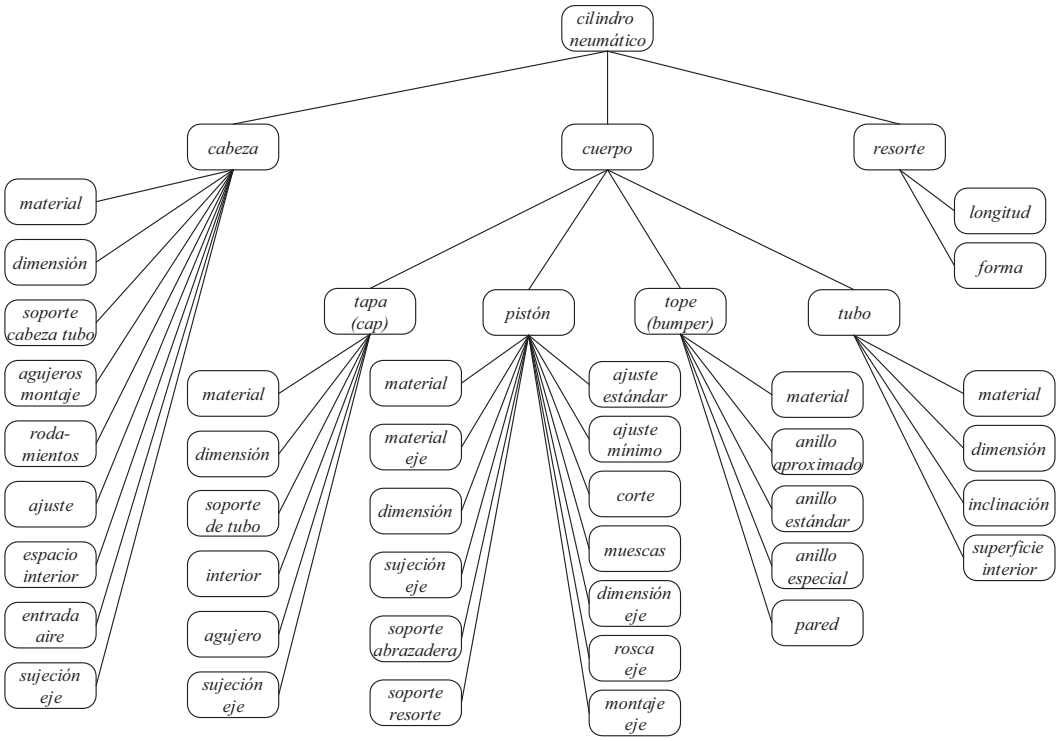


Figura 6.18: Jerarquía de especialistas del sistema AIR-CYL.

Los especialistas de niveles más altos tratan con aspectos más generales de los componentes y los de niveles más bajos con componentes más específicos con decisiones de diseño de más detalle. El especialista raíz es responsable del diseño del sistema completo. Todos los especialistas pueden acceder a una base de datos común que mantiene el diseño conforme se va desarrollando. Sobre la base los especialistas

hacen consultas, modificaciones y comprobación de si se verifican determinadas restricciones. A nivel básico se tienen especialistas elementales (llamados *tareas* en la denominación original de DSPL) que captura conocimiento de diseño local.

La figura 6.18 muestra la jerarquía de especialistas que maneja el sistema AIR-CYL. El nodo raíz representa el especialista global que toma decisiones de diseño sobre el sistema completo. Dicho especialista delega decisiones de diseño en otros tres especialistas, cabeza, cuerpo y resorte, que corresponden a las partes principales del sistema a diseñar. A su vez, dichos nodos pueden delegar sus decisiones en otros tal como muestra la jerarquía. Algunos especialistas de bajo nivel se encargan de decisiones de diseño relativamente sencillas como determinar el material del componente o sus dimensiones. Sobre un mismo tipo de decisión puede haber especialistas alternativos que aplican criterios diferentes (ajuste estándar y mínimo, o anillo estándar y especial).

Para describir cada especialista se hace uso del lenguaje DSPL que permite formular el conocimiento de cada especialista con una notación funcional con un estilo similar al lenguaje LISP, lenguaje en el que estaba implementado DSPL. Por ejemplo, la descripción del especialista *bumper* en DSPL es la siguiente:

```
(SPECIALIST
  (NAME Bumper)
  (USED-BY Rest)
  (USES None)
  (DESIGN-PLANS BumperDP1 BumperDP2)
  (ROUGH-DESIGN-PLANS BumperRDP1))
```

Con ello se indica que el especialista *Bumper* (tope del cilindro) es utilizado por otro especialista llamado *Rest* (el especialista en el cuerpo del cilindro). No utiliza a especialistas de nivel intermedio y tiene tres planes de diseño, de los cuales dos de ellos son normales y uno es aproximado (*rough*). Aunque este ejemplo no lo muestra, el lenguaje permite indicar dentro de cada especialista restricciones de aplicabilidad

(condiciones para que se pueda aplicar el especialista) y restricciones de validación del resultado que se verifican tras las decisiones de diseño.

Los *planes de diseño* en un especialista representan métodos para diseñar la parte, área o función sobre la que actúa el especialista. Los planes de un especialista indican las distintas formas de realizar esa parte del diseño, expresados todos al mismo nivel de detalle. El plan está formado por una secuencia (lineal o en paralelo) de: (1) especialistas que refinan diseño, (2) especialistas de detalle (llamados tareas) que deciden sobre los valores de parámetros del diseño y (3) restricciones que aseguran la integridad de las decisiones.

```
(PLAN
  (NAME BumperRDP1)
  (TYPE RoughDesign)
  (USED-BY Bumper)
  (USES None)
  (QUALITIES )
  (HISTORY None)
  (INITIAL-CONSTRAINTS None)
  (FINAL-CONSTRAINTS None)
  (BODY
    BumperMaterial
    BumperFlangeRough))

(TASK
  (NAME BumperFlangeRough)
  (USED-BY BumperRDP1)
  (INITIAL-CONSTRAINTS None)
  (FINAL-CONSTRAINTS None)
  (BODY BumperFlangeThicknessRough))

(STEP
  (NAME BumperFlangeThicknessRough)
  (ATTRIBUTE-NAME BumperFlangeThickness)
  (USED-BY BumperFlangeRough)
  (BODY
    (KNOWN
      EnvelopeLength  (KB-FETCH 'Requirements 'EnvelopeLength)
      Factor          0.0077)
    (DECISIONS
      Thickness       (TWO-DP (* Factor EnvelopeLength))
      REPLY           (KB-STORE 'Bumper 'BumperFlangeThickness Thickness))))
```

Figura 6.19: Ejemplo de formulación de un plan de AIR-CYL

Los planes pueden ser normales o aproximados. Los *planes aproximados* permiten decidir y comprobar la coherencia de los valores de los que gran parte del diseño depende. Ello permite (1) darse cuenta rápidamente de si se puede descartar el diseño por imposible antes de entrar en los detalles, y (2) podar la búsqueda puesto que fija

una serie de valores principales de los cuales dependen otros valores. Además, en los casos en que existen valores mutuamente dependientes, los planes aproximados fijan el valor provisional de uno de ellos a partir del cual se derivarán otras decisiones.

Todos los planes (normales y aproximados) se formulan con la misma sintaxis. La figura 6.19 muestra un ejemplo de formulación de plan. La parte principal está en el cuerpo (*body*) que indica las acciones que lo forman, en este caso dos: `bumperMaterial` y `bumperFlangeRough`.

Obsérvese que en el cuerpo del plan se invoca a otro elemento (una tarea) que representa un especialista de bajo nivel. A su vez, en el cuerpo de la tarea se invoca a otro elemento denominado *paso de diseño* (*step*) que corresponde a una decisión elemental (por ejemplo elegir un material, fijar una dimensión, etc.). La decisión del paso de diseño es tentativa, es decir, puede más adelante violar alguna restricción y como consecuenca ser modificada. En el ejemplo de la figura, la decisión tiene como fin determinar el valor inicial del grosor del anillo haciendo el cálculo `0.0077*EnvelopeLenght`. Se consultan valores a la base de datos que mantiene el diseño en curso con `KB-FETCH` y se actualizan con `KB-STORE`. En general, los pasos de diseño pueden incluir restricciones como muestra el siguiente ejemplo (penúltima línea), en donde se comprueba que el valor de un parámetro es superior a otro:

```
(STEP
  (NAME CapBackFaceThickness)
  (ATTRIBUTE-NAME CapBackFaceThickness)
  (USED-BY CapBackFace)
  (REDESIGN NOT-POSSIBLE)
  (FAILURE-SUGGESTIONS
    (SUGGEST (INCREASE CapDepth))
    (SUGGEST (DECREASE CapInternalDepth)) )
  (BODY
    (KNOWN
      CapDepth        (KB-FETCH 'Cap 'CapDepth)
      InternalDepth   (KB-FETCH 'Cap 'CapInternalDepth))
    (DECISIONS
      Thickness       (- CapDepth InternalDepth)
      REPLY           (TEST-CONSTRAINT CapWall>MinTh)
      REPLY           (KB-STORE 'Cap 'CapBackFaceThickness Thickness))))
```

En caso de que no se verifique la restricción, DSPL da la posibilidad de indicar qué acción puede resolverla. En el ejemplo anterior obsérvese que se dan dos sugerencias de cómo resolver la violación de la restricción mediante incremento (`INCREASE`) de un parámetro o decremento (`DECREASE`) de otro parámetro. Cuando en DSPL no se indica de forma explícita la medida de incremento o decremento, tal como es este caso, se toma un valor mínimo prefijado.

La forma de realizar inferencia sigue un enfoque similar al enfoque general de configuración jerárquica HTN con alguna variante. El proceso parte de una acción de diseño global expresada por especialista raíz, seleccionando planes que se refinan con la intervención de especialistas de más detalle. Los planes son tentativos y pueden fallar aunque no se esperan muchos fallos en su realización dado que son resultado de la experiencia en la resolución manual de problemas de diseño. Cuando un plan falla (por alguna de sus acciones) puede haber dos formas de actuar. Una forma es eligiendo otro plan (solución que sigue el método general de configuración jerárquica HTN). La otra solución consiste en resolver el fallo llevando a cabo las sugerencias para remediar la violación de una restricción de acuerdo con la estrategia LBBS (*Least Backup by Suggestion*) que es similar a la forma en que se resuelven violaciones en el método de proponer-y-revisar. En el caso de AIR-CYL las sugerencias se tratan de una en una aunque, como reconocen sus autores, es posible que en determinados dominios se deba contemplar alguna forma de realizar combinaciones, aspecto no considerado en DSPL.

Tras el fallo de un plan se realiza de nuevo una selección en donde planes que inicialmente eran candidatos pueden dejar de serlo debido a que se haya indicado explícitamente en el conjunto de criterios de selección. Se trata de relaciones que indican que si el plan *A* falla entonces no vale la pena intentar el plan *B*, lo que puede ahorrar esfuerzo de búsqueda en ciertos casos.

La inferencia de AIR-CYL primero hace una búsqueda por los especialistas aplicando sólo planes aproximados con lo que se fijan valores esenciales del diseño. Después se hace otra búsqueda con planes normales para completar el diseño. Normalmente en la primera búsqueda se fijan los materiales y algunas dimensiones principales.

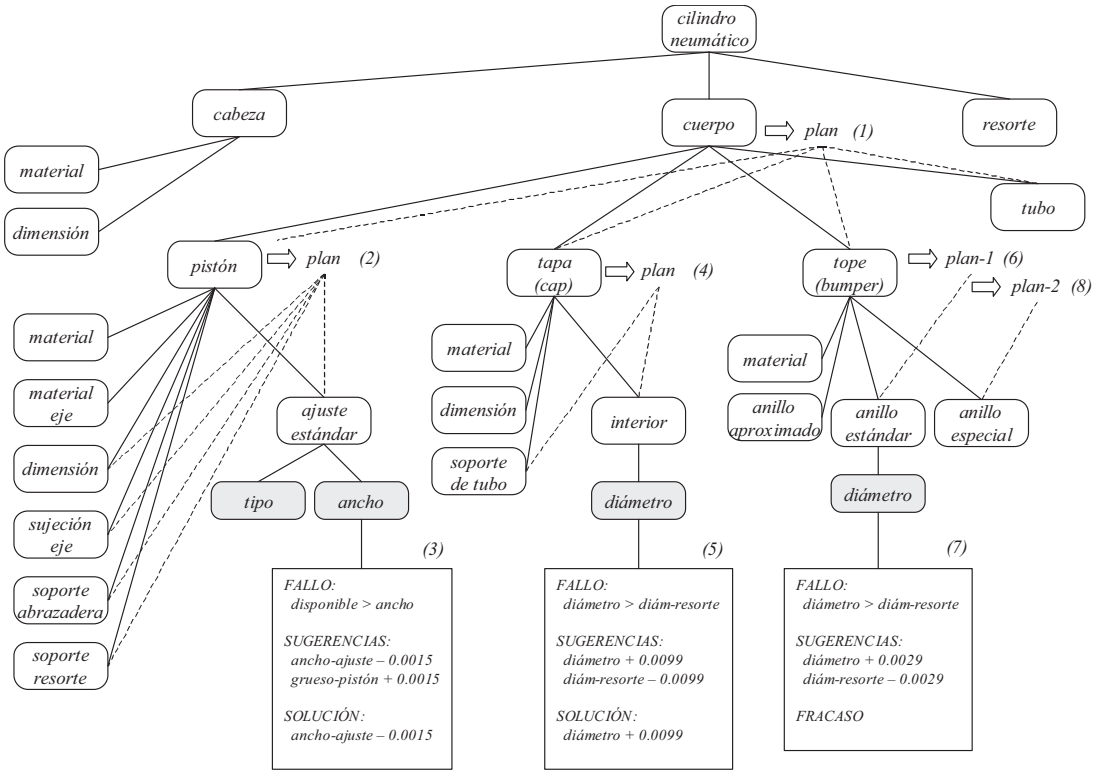


Figura 6.20: Ejemplo de búsqueda realizada por AIR-CYL.

La figura 6.20 resume un ejemplo de la forma en que se realiza la búsqueda en AIR-CYL. Se muestra sólo una parte de la búsqueda (a partir del especialista cuerpo del cilindro) señalando con números entre paréntesis el orden en que se realizan los diferentes etapas. Las acciones realizadas son las siguientes:

(1) El especialista cuerpo del cilindro genera un plan posible que invoca a los especialistas piston, cap, bumper y tubo en ese orden.

(2) El especialista pistón genera un plan con una serie de especialistas (dimensión, sujección eje, soporte abrazadera, etc.) a los que invoca en secuencia.

(3) Cuando llega al especialista ajuste estándar, se realizan dos pasos de diseño: uno para elegir el tipo de ajuste y otro para decidir el ancho. Cuando se llega a la decisión del ancho la decisión inicial basada en ciertos criterios no cumple una restricción debido a que el tamaño del pistón no es suficiente para el ancho de ajuste previsto. Se sugieren dos alternativas para resolverlo de las cuales funciona la primera. El plan continúa con otros especialistas (no mostrados en la figura) hasta que se completa con éxito.

(4) El especialista de la tapa del cilindro genera un plan que invoca a ciertos especialistas (soporte de tubo, etc.).

(5) Cuando se alcanza el especialista en el interior, éste invoca a un paso de diseño para decidir el diámetro. La decisión inicial basada en ciertos parámetros no satisface la restricción de que debe ser mayor que el diámetro del resorte. Para resolverlo se proponen dos cambios y uno de ellos funciona con éxito. El plan continúa con otros sub-especialistas no indicados en la figura hasta que termina con éxito.

(6) El especialista del tope del cilindro genera un plan (denominado en la figura como plan-1). Dicho plan invoca al especialista anillo estándar.

(7) Cuando se trata de determinar el diámetro del anillo de acuerdo con el procedimiento estándar se viola una restricción sobre la que después no es posible aplicar soluciones. Por tanto se produce un fallo al refinar plan-1.

(8) Se busca otro plan alternativo en el especialista tope del cilindro que genera otra opción denominada plan-2. Esta opción invoca a un especialista diferente en la forma de determinar el anillo. A partir de ahí la búsqueda continúa con otros especialistas.

En resumen, el sistema AIR-CYL supone la aplicación y desarrollo de las ideas presentadas en la descripción del método general de configuración jerárquica HTN incluyendo como aspectos particulares (1) el manejo de especialistas que es en cierta forma equivalente a las acciones de diseño de la configuración HTN, (2) la forma en que se toman decisiones sobre diseño en cada nodo de la jerarquía en donde una propuesta inicial puede ser revisada con ayuda de sugerencias recogidas de forma explícita al estilo del método de proponer y revisar, (3) la realización de un primer proceso de búsqueda para decisiones preliminares con planes aproximados seguido de un segundo paso que refina el proceso.

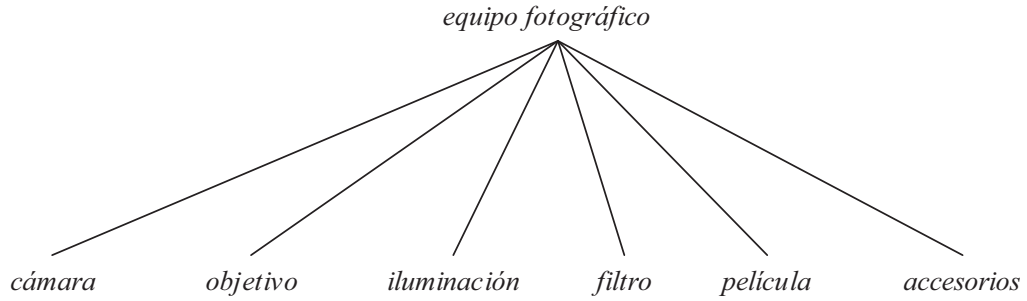


Figura 6.21: Especialistas considerados en el problema de configuración de equipos fotográficos.

6.4.3. Ejemplo 3: Venta de equipos fotográficos

Un ejemplo resoluble mediante el método estudiado en este capítulo es el problema de venta de equipos configurables. En este capítulo se ilustra parcialmente este caso considerando una organización de decisiones de diseño, tal como se realiza en el ejemplo anterior, para el dominio de configuración de equipos fotográficos a partir de las necesidades y preferencias de clientes [Molina, 01a; Molina 01b; Serna, 99].

En este problema se tiene un acción de diseño raíz encargada de la decisión global de la configuración del equipo fotográfico. En un siguiente nivel se encuentran las acciones de diseño de los diferentes componentes del equipo fotográfico: (1) cámara, para el cuerpo de la cámara, (2) objetivo, para el objetivo de la cámara, (3) iluminación, referente a tipos de flash o focos adicionales, (4) filtro, (5) película y (6) accesorios (figura 6.21). Al igual que el caso AIR-CYL se denominarán especialistas a dichas acciones de diseño. El razonamiento en cada uno de dichos especialistas es de tipo clasificativo y consiste en seleccionar dentro de un conjunto prefijado de soluciones la que más se ajusta al cliente teniendo en cuenta, además, las decisiones que han sido tomadas para los otros especialistas. La figura 6.22 muestra ejemplos de las soluciones que debe determinar cada tipo de especialista.

Aunque en el presente texto se ha indicado de forma general para el método de configuración jerárquica HTN una forma sencilla con reglas para representar el conocimiento, la construcción de un sistema en un problema de dimensión real puede requerir utilizar dentro de cada especialista formas de representación más ricas y/o complejas. Así, en el caso del especialista de selección de cámaras se puede optar por una representación en forma de marcos, organizados en una jerarquía de tipos de cámaras (compacta, panorámica, digital, etc.). En cada marco se dispone de *slots* relativos a necesidades y preferencias del cliente. La equiparación de los datos concretos de un determinado cliente con dichos *slots* de acuerdo con el conveniente

uso de conocimiento de control (qué indica condiciones mínimas y/o máximas de equiparación) dará lugar a la selección de un tipo de cámara concreto.

Especialista	cámara	objetivo	iluminación	filtro	película	accesorios
Soluciones posibles	Réflex paso universal Reflex medio formato Réflex arrastre Compacta Compacta zoom Aps varios usos Panorámica Digital Gran formato Sumergible ...	Objetivo 50 mm Objetivo 90 mm Objetivo 20 mm Zoom 70-210 mm Zoom 28-80 mm Teleobjetivo corto Teleobjetivo largo Gran ang. 28 mm ...	Flash antorcha Flash anular Flash electrónico Luz cenital Foco directo ...	Normal Polarizador Calentador Gris neutro Paisajes Tabaco claro Tabaco oscuro Densidad ...	BN normal 9x13 BN alta 9x13 BN normal 35 BN alta 35 Color normal 9x13 Color alta 35 ...	Trípode Monopie Fotómetro Funda Parasol Baterías Disparador ...

Figura 6.22: Ejemplos de soluciones posibles para cada uno de los especialistas del ejemplo.

Un sistema que utiliza este método puede operar en la recomendación de venta de equipos fotográficos simulando cómo razona un vendedor en dicho dominio. Como resultado del razonamiento realizado por dicho método se obtiene una configuración formada por un tipo de cámara (por ejemplo, réflex medio formato), un objetivo (por ejemplo 50 mm), etc. A continuación, realizando una búsqueda de forma separada del método, pueden seleccionarse componentes de determinadas firmas haciendo de forma automática una consulta a una base de datos que disponga de dicha

información, simulando cómo el vendedor busca los componentes de determinados tipos en un cierto catálogo. Finalmente, para filtrar las diversas opciones encontradas en dicha búsqueda pueden aplicarse criterios tales como: precio, calidad, preferencias comerciales (stock, margen de beneficio, comisiones, antigüedad), etc.

6.4.4. Ejemplos de generalización del método planificación jerárquica

La descomposición de un problema en subproblemas de acuerdo con la organización en planes de acciones es un enfoque poderoso a la vez que intuitivo y puede generalizarse a problemas diferentes de planificación y configuración en donde las acciones de niveles más bajos resuelven diferentes tipos de tareas con formas distintas de representación. La idea básica consiste en que la forma de descomponer problemas se expresa mediante planes que, en este caso, son planes más generales (de resolución de partes del problema) y que se seleccionan de forma heurística de acuerdo con las características de cada caso. En cada acción (global o elemental) puede haber bloqueos lo que obliga a desarrollar una búsqueda que puede reconsiderar planes anteriores. Las conclusiones alcanzadas mediante acciones previas condicionan a las posteriores. Los siguientes ejemplos ilustran esta idea:

- *Asignación de guardias*. El ejercicio 5.5 muestra un caso de problema de asignación de guardias de servicios hospitalarios que se descompone en dos subproblemas: uno de asignación de guardias a médicos residentes y otro de asignación a médicos adjuntos. Las conclusiones alcanzadas por el primero condicionan la asignación del segundo y los fallos del segundo pueden hacer reconsiderar al primero. Esto puede verse como dos acciones de asignación que resuelven partes del problema de asignación. El problema podría extenderse para considerar otros miembros del servicio de guardias (ATS, auxiliares, etc.) con otras acciones de asignación. Además puede haber acciones de nivel superior de forma que, en función de determinadas consideraciones, eligiera formar equipos con un determinado número de

personas en cada guardia. Por ejemplo, en un hospital en zona turística en verano los servicios hospitalarios deben tener más personas. En este caso se tienen *planes de asignación* que se deciden y completan razonando en un descenso jerárquico.

- *Predicción*. El problema de predicción puede considerarse también bajo este enfoque. Por ejemplo, considérese que se tiene una acción de predicción que aplica un método de clasificación heurística para predecir el estado de un sistema en el siguiente instante de tiempo. Esta acción de predicción puede ser manejada por otra de nivel superior que utiliza planes de predicción para invocarla de forma iterativa para obtener predicciones para instantes de tiempos consecutivos que forman una predicción global. Estos planes podrían tener la estructura de:

  ```
  P = {predicción(T1), predicción(T2), predicción(T3)}
  ```

  En donde `predicción(Ti)` es una acción de predicción para el instante `Ti`. Lógicamente, las predicciones realizadas en los instantes previos condicionan a las futuras.

Como caso general, los efectos de las acciones concretas pueden considerarse de forma heterogénea, es decir, pueden invocar a procedimientos específicos que realizan operaciones mediante otros métodos de resolución de problemas o con algoritmos convencionales (búsquedas en bases de datos, búsquedas en internet, procedimientos estadísticos, etc.). Por tanto, el método de planificación jerárquica descrito en este capítulo puede concebirse como un mecanismo flexible de control heurístico para integración de procesos locales de resolución de problemas y algoritmos de diversa naturaleza.

6.5 Ejercicios

EJERCICIO 6.1. Supóngase que se tiene un modelo para ayuda a la decisión en problemas de transporte público (líneas de autobuses urbanos) formulado de acuerdo con el método de *planificación jerárquica HTN*. El conocimiento relativo a estrategias incluye las siguientes sentencias:

```
gestionar-red → gestionar-en-orden-prioridad


gestionar-línea(X) y retraso(X)=individual y causa(X)=(demanda o tráfico)
                   y gravedad(X)=alta → reducir-recorrido(X)
gestionar-línea(X) y retraso(X)=individual
                   y causa(X)=demanda y gravedad(X)=alta → reforzar-línea
gestionar-línea(X) y retraso(X)=individual y causa(X)=corte  → variar-recorrido(X)
gestionar-línea(X) y retraso(X)=general y causa(X)=no(avería) → cambiar-horario(X)
gestionar-línea(X) y causa(X)=avería y gravedad(X)=alta → asistencia-con-refuerzo(X)
gestionar-línea(X) y causa(X)=avería y gravedad(X)=leve → asistencia-sin-refuerzo(X)


asignar-reserva(L) y pertenece(X)=L y estado(X)=retrasado y tipo(Y)=reserva
                   y disponible(Y)=sí y zona(X)=final → refuerzo-desde-inicio(L,X,Y)
asignar-reserva(L) y pertenece(X)=L y estado(X)=retrasado y tipo(Y)=reserva
                   y disponible(Y)=sí y recorrido(X)=completo
                   y zona(X)=no(final) → refuerzo-desde-final(L,X,Y)
asignar-reserva(L) y pertenece(X)=L y estado(X)=retrasado y disponible(Y)=sí
                   y recorrido(X)=parcial
                   y próxima-parada(X)=Z → refuerzo-desde-parada(L,X,Y,Z)
asignar-reserva(L) y causa(L)=avería y pertenece(X)=L y estado(X)=retrasado
                   y tipo(Y)=reserva
                   y disponible(Y)=sí → refuerzo-desde-inicio(L,X,Y)
```

```
modificar-recorrido(L) y pertenece(X)=L y estado(X)=retrasado y parada(X)=Y
                       y giro-posible(Y)=sí y causa(L)=no(corte)
                       y zona(X)=no(final) → limitar-recorrido(L,X,Y)
modificar-recorrido(L) y pertenece(X)=L y estado(X)=retrasado y parada-actual(X)=Y
                       y giro-posible(Y)=no  y causa(L)=no(corte)
                       y zona(X)=no(final) → saltar-paradas(L,X,Y)
modificar-recorrido(L) y causa(L)=corte y parada-afectada(L)=P y
                       y alternativo(P)=X → recorrido-alternativo(L,X)

rotar-horario(L) y gravedad(L) = leve → rotación(simple)
rotar-horario(L) y gravedad(L) = alta → rotación(compuesta)
```

Respecto al conocimiento sobre acciones se tiene el siguiente modelo:

```
gestionar-en-orden-prioridad   → {gestionar-línea(línea-1),
                                  gestionar-línea(línea-2),
                                  gestionar-línea(línea-3)}
reforzar-linea(L)              → {asignar-reserva(L)}
reducir-recorrido(L)           → {modificar-recorrido(L), asignar-reserva(L)}
variar-recorrido(L)            → {modificar-recorrido(L)}
cambiar-horario(L)             → {rotar-horario(L)}
asistencia-con-refuerzo(L)     → {ordenar-asistencia-técnica(L), asignar-reserva(L)}
asistencia-sin-refuerzo(L)     → {ordenar-asistencia-técnica(L)}
refuerzo-desde-inicio(L,X,Y)   → {ordenar-servicio(Y,L, parada-inicio),
                                   notificar-refuerzo(X)}
refuerzo-desde-final(L,X,Y)    → {ordenar-servicio(Y,L, parada-final),
                                   notificar-refuerzo(X)}
refuerzo-desde-parada(L,X,Y,Z) → {ordenar-servicio(Y,L,Z), notificar-refuerzo(X)}
limitar-recorrido (L,X,Y)      → {alcanzar-parada(X,Y), transferir-pasajeros(X),
                                   cambiar-sentido(X)}
```

```
saltar-paradas(L,X)            → {ordenar-salto-paradas(X)}
recorrido-alternativo(L,X)     → {ordenar-alternativo(L,X)}
rotación(L)                    → {ordenar-rotación(L)}
```

Los efectos de las acciones concretas son los siguientes:

```
ordenar-servicio(X,L,P) y disponible(X)=sí → disponible(X)=no
cambiar-sentido(X) y pertenece(X,L) → recorrido(L)= parcial
```

SE PIDE:

1. Teniendo en cuenta que, en un determinado momento, la línea-1 presenta un retraso individual, el autobús retrasado es el A19 (*retrasado = A19*), la gravedad es alta, la zona es inicial, el autobús de reserva está disponible y el identificador de dicho autobús es el A12 (*refuerzo = A12*), aplicar el método de planificación jerárquica HTN para obtener un plan dirigido a resolver el problema de gestionar la línea-1 (no se deben tener en cuenta en este caso línea-2 ni línea-3). Si se pregunta durante el proceso de desarrollo, debe considerarse que *giro = posible*, *causa=demanda*, *parada = P20* y *recorrido = completo*. Detallar los pasos realizados y dibujar el gráfico que muestra el proceso. Detener el proceso una vez que se encuentra una solución.

2. ¿Cuál sería el resultado si en vez de *zona = inicial* fuera *zona = final*?

EJERCICIO 6.2. Considérese la siguiente aplicación del método de *configuración jerárquica HTN* a un problema simplificado de configuración de viajes turísticos. En este problema, a partir de las características y preferencias de un cliente se determinan las características de un viaje turístico que puede satisfacer las preferencias de dicho cliente. El resultado obtenido es una descripción general de las

características del viaje (no un plan del recorrido) por lo que puede considerarse como un problema de configuración. El conocimiento sobre estrategias de configuración del viaje es el siguiente:

Acción	**Condiciones de selección de estrategia**
configurar-vacaciones	edad > 30 y acompañante = pareja → viaje-crucero edad > 25 y acompañante = pareja o familia → visita-cultural acompañante = pareja o familia → estancia-playa edad < 35 y acompañante = no(familia) → viaje-aventura edad < 29 y acompañante = grupo → estancia-rural
elegir-tipo-crucero	disponibilidad-económica = no(baja) y duración >= 2 semanas → crucero(Caribe) disponibilidad-económica = no(baja) y época-año = verano → crucero(Atlántico) disponibilidad-económica = baja y época-año = verano → crucero(Mediterráneo)
elegir-destino	tipo-viaje = playa y época-año = invierno → destino(Cuba) zona-destino = Caribe → destino(Cuba) tipo-viaje = cultural → destino(Norte-Europa) zona-destino = Atlántico → destino(Norte-Europa) tipo-viaje = cultural → destino(Grecia) tipo-viaje = aventura y disponibilidad-económica = no(baja) → destino(Centroamérica) tipo-viaje = playa y época-año = verano → destino(Levante) tipo-viaje = rural → destino(Castilla) zona-destino = Mediterráneo → destino(Barcelona)
elegir-transporte	zona-destino = Caribe → vuelo(atlántico) destino = Cuba o Centroamérica → vuelo(atlántico) destino = Barcelona o Levante o Castilla → terrestre destino = Barcelona o Levante → vuelo(nacional) destino = Norte-Europa o Grecia → vuelo(europeo)
elegir-línea-aérea(X)	disponibilidad-económica = alta → línea-aérea(Iberia) X = atlántico y disponibilidad-económica=media y anticipación-compra=larga → línea-aérea(Iberia) (X = nacional o europeo) y disponibilidad-económica = media → línea-aérea(Iberia) tipo-vuelo = nacional o Europa → línea-aérea(Spanair) tipo-vuelo = nacional o Europa → línea-aérea(Europa)
elegir-medio-terrestre	disponibilidad-económica = alta → transporte-terrestre(tren-rápido) disponibilidad-económica = baja → transporte-terrestre(tren-económico) disponibilidad-económica = baja → transporte-terrestre(autobús)
elegir-alojamiento	disponibilidad-económica = alta y destino = no(Norte-Europa) → alojamiento(hotel-lujo) disponibilidad-económica = media o alta → alojamiento(hotel-normal) disponibilidad-económica = baja → alojamiento(albergue)
elegir-visita-organizada	disponibilidad-económica = alta → visita-completa disponibilidad-económica = no(alta) → visita-económica

Se dispone de conocimiento de acciones formado por las siguientes sentencias:

```
viaje-crucero     → {elegir-tipo-crucero, elegir-destino, elegir-transporte}
estancia-playa    → {fijar-tipo-viaje(playa),
                     elegir-destino, elegir-transporte, elegir-alojamiento}
estancia-rural    → {fijar-tipo-viaje(rural),
                     elegir-destino, elegir-transporte, elegir-alojamiento}
viaje-aventura    → {fijar-tipo-viaje(aventura), elegir-destino, elegir-transporte,
                     elegir-alojamiento}
visita-cultural   → {fijar-tipo-viaje(cultural), elegir-destino, elegir-transporte,
                     elegir-alojamiento, elegir-visita-organizada}
crucero(X)        → {fijar-tipo-viaje(crucero),  fijar-zona-destino(X)}
destino(X)        → {fijar-destino(X)}
vuelo(X)          → {elegir-línea-aérea(X)}
linea-aérea(X)    → {fijar-linea-aérea(X)}
terrestre         → {elegir-medio-terrestre}
transporte-terrestre(X) → {fijar-terrestre(X)}
alojamiento(X)    → {fijar-alojamiento(X)}
visita-completa   → {fijar-visita(sí), fijar-guía(personal)}
visita-económica  → {fijar-visita(sí), fijar-guía(grupo)}
```

La base de conocimiento de efectos es la siguiente:

```
fijar-tipo-viaje(X)    → tipo-viaje = X
fijar-zona-destino(X)  → zona-destino = X
fijar-destino(X)       → destino = X
fijar-linea-aérea(X)   → línea-aérea = X
fijar-terrestre(X)     → transporte-terrestre = X
fijar-alojamiento(X)   → alojamiento = X
fijar-visita(X)        → visita-organizada = X
fijar-guía(X)          → guía-turístico = X
```

SE PIDE:

1. Aplicar el método de *configuración jerárquica HTN* a este problema para obtener el viaje recomendable para un cliente con los siguientes datos: *edad=36, acompañante=pareja, época-año=invierno, anticipación-compra=corta, disponibilidad-económica=media, duración=3 semanas*. Explicar los pasos seguidos en el proceso de resolución y dibujar el árbol de

búsqueda que realiza el proceso. Obtener únicamente la primera solución válida mostrando claramente el conjunto de conclusiones deducidas.

2. Explicar brevemente, apoyándose en árboles de búsqueda, cómo continuaría el proceso de resolución del problema para encontrar otras soluciones válidas para el mismo caso del apartado anterior. No es necesario en este apartado explicar los detalles de cada paso realizado.

3. Considerar otro caso diferente al anterior en donde los datos son: *edad=18, acompañante=grupo, disponibilidad-económica=baja*. Obtener todas las soluciones válidas. Para este apartado es suficiente realizar una resolución gráfica mostrando árboles de búsqueda. Indicar claramente los hechos deducidos.

NOTA: Ante varias opciones durante el proceso de búsqueda, elegir primero las opciones que están antes según el orden que presentan en el enunciado.

EJERCICIO 6.3. Para resolver un problema de decisión sobre inversiones se dispone de un modelo basado en el método de *planificación jerárquica HTN*. En este problema, a partir de los datos sobre un cliente y otros sobre contexto económico, se trata de determinar cuál es la opción u opciones de inversión más adecuadas para dicho cliente. Para ello se manejan los siguientes atributos:

Atributo	**Significado**
`cantidad`	Cantidad que el cliente desea invertir expresada en euros.
`plazo`	Duración de la inversión expresada en años.
`edad`	Edad del cliente en años.
`cargas`	Indica si el cliente tiene cargas (por ej. de tipo familiar, deudas, etc.). Valores posibles: {sí, no}
`modalidad`	Modalidad de la inversión. Valores posibles: {convencional, diversificada}
`perfil`	Perfil del cliente. Valores posibles: {conservador, moderado, arriesgado}
`ciclo`	Situación del ciclo económico. Valores posibles: {descendiente, estable, ascendiente}
`divisa`	Relación euro-dólar. Valores posibles: {favorable, desfavorable}

El conocimiento de planificación incluye las siguientes estrategias:

Acción	**Condiciones de selección de estrategias**
invertir	cantidad = X y X < 50.000 y plazo < 10 → pequeña-inversión(X) cantidad = X y X < 50.000 y plazo >= 10 → pequeña-inversión-largo-plazo(X) cantidad = X y X >= 50.000 y X < 150.000 → inversión-media(X) cantidad = X y X >= 150.000 y plazo < 10 → gran-inversión-corto-plazo(X) cantidad = X y X >= 150.000 y plazo >= 10 y edad < 60 → gran-inversión-convencional(X) cantidad = X y X >= 150.000 y plazo >= 10 y edad >= 60 → gran-inversión-conservadora(X) cantidad = X y X >= 150.000 y plazo >= 10 → gran-inversión-diversificada(X)
invertir-en-monetaria(X)	edad < 35 → perfil = arriesgado edad >= 35 y edad =< 55 y cargas = no → perfil = moderado edad >= 35 y edad =< 55 y cargas = sí → perfil = conservador edad >= 55 → perfil = conservador --- plazo =< 1 → depósito-bancario(X) plazo >1 y plazo =< 20 y perfil = NO(arriesgado) y ciclo = NO(creciente) → fondo-renta-fija(X) plazo >1 y plazo =< 20 y perfil = NO(conservador) → fondo-mixto(X) plazo >1 y plazo =< 20 y perfil = moderado y ciclo =creciente → fondo-renta-variable(X) plazo > 1 y plazo =< 20 y perfil = arriesgado → fondo-renta-variable(X) plazo > 20 → fondo-pensiones(X)
invertir-en-inmuebles(X)	X >= 1.000.000 → local-comercial(X) X >= 1.000.000 → vivienda-lujo(X) X >= 1.000.000 → inmueble-diversificado(X) modalidad = convencional y X < 300.000 → inmueble-bajo-coste(X) modalidad = diversificada y X < 500.000 → inmueble-bajo-coste(X) X >= 300.000 y X < 1.000.000 → inmueble-medio(X)
invertir-en-tangibles(X)	modalidad = convencional y X >= 75.000 → tangible-gama-alta(X) modalidad = diversificada y X >= 10.000 → tangible-gama-alta(X) modalidad = diversificada y X >= 10.000 → tangible-diversificado(X) modalidad = diversificada y X >= 5.000 y X < 10.000 → tangible-gama-media(X) modalidad = diversificada y X < 5.000 → tangible-gama-baja(X) modalidad = diversificada y X < 5.000 → tangible-gama-baja(X)

Junto al conocimiento anterior sobre estrategias, se dispone de conocimiento de acciones formado por las siguientes sentencias:

```
pequeña-inversión(X)              → {invertir-en-monetaria(X)}
pequeña-inversión-largo-plazo(X)  → {invertir-en-monetaria(0.5*X),
                                     invertir-en-tangibles(0.5*X)}
inversión-media(X)                → {invertir-en-monetaria(0.75*X),
                                     invertir-en-tangibles(0.25*X)}
gran-inversión-corto-plazo(X)     → {invertir-en-monetaria(X)}
gran-inversión-convencional(X)    → {fijar-modalidad(convencional),
                                     invertir-en-inmuebles(0.7*X),
                                     invertir-en-monetaria(0.2*X),
                                     invertir-en-tangibles(0.1*X)}
gran-inversión-diversificada(X)   → {fijar-modalidad(diversificada),
                                     invertir-en-inmuebles(0.7*X),
                                     invertir-en-monetaria(0.2*X),
                                     invertir-en-tangibles(0.1*X)}
gran-inversión-conservadora(X)    → {invertir-en-inmuebles(0.5*X),
                                     invertir-en-monetaria(0.5*X)}
gran-inmueble(X)                  → {local-comercial(X)}
gran-inmueble(X)                  → {vivienda-lujo(X)}
inmueble-diversificado(X)         → {vivienda-ciudad(0.8*X),plazas-garaje(0.2*X)}
inmueble-medio(X)                 → {vivienda-ciudad(X)}
inmueble-medio(X)                 → {apartamento-playa(X)}
inmueble-bajo-coste(X)            → {plazas-garaje(X)}
tangible-gama-alta(X)             → {obra-de-arte(X)}
tangible-diversificado(X)         → {joyas(0.75*X),numismática(0.25*X)}
tangible-gama-media(X)            → {joyas(X)}
tangible-gama-baja(X)             → {numismática(X)}
tangible-gama-baja(X)             → {filatelia(X)}
fondo-renta-fija(X)               → {letras-del-tesoro(X)}
fondo-renta-fija(X)               → {bonos-convertibles(X)}
fondo-mixto(X)                    → {fondo-mixto-conservador(X)}
fondo-mixto(X)                    → {fondo-mixto-arriesgado(X)}
fondo-renta-variable(X)           → {bolsa-Europa(X)}
fondo-renta-variable(X)           → {bolsa-EEUU(X)}
fondo-renta-variable(X)           → {bolsa-Japón(X)}
```

Por ejemplo, la tercera sentencia indica que el esquema *inversión-media(X)*, con el parámetro *X* correspondiente a la cantidad a invertir, se descompone en dos acciones abstractas: (1) *invertir-en-monetaria(0.75*X)* que indica realizar una inversión monetaria con el 75% de la cantidad a invertir y (2) *invertir-en-tangibles(0.25*X)* que indica realizar una inversión en bienes tangibles con el 25% de la cantidad a invertir.

La base de conocimiento de *efectos* tiene las siguientes sentencias:

```
fijar-modalidad(X) → modalidad = X
letras-del-tesoro(X) y ciclo = descendiente
bonos-convertibles(X) y ciclo = estable
fondo-mixto-conservador(X) y ciclo = descendiente
fondo-mixto-arriesgado(X) y ciclo = creciente
bolsa-Europa(X) y divisa = favorable
bolsa-EEUU(X) y divisa = desfavorable y ciclo = NO(creciente)
bolsa-Japón(X) y divisa = desfavorable y ciclo = creciente
```

La solución al problema se representa con un conjunto de acciones de inversión en donde cada uno de ellas tiene asociado entre paréntesis la cantidad a invertir. Por ejemplo, una solución podría ser el siguiente plan de inversión:

```
Plan = {vivienda-ciudad(325.000),plazas-garaje(10.000),
       letras-del-tesoro(25.000),obras-de-arte(25.000)}
```

Dicho plan, aunque se indica mediante un conjunto de acciones con un determinado orden, se puede ejecutar sin importar el orden debido a las características propias del problema.

SE PIDE:

1. Aplicar el método de *planificación jerárquica HTN* a este problema para obtener una recomendación sobre inversión sabiendo que se tienen como datos iniciales: *cantidad = 500.000 €, plazo = 15 años, edad = 43 años.* Si a lo largo del proceso se solicita información adicional considerar los siguientes valores: *cargas = no, ciclo = creciente, divisa = favorable*. Explicar los pasos seguidos en el proceso de resolución y dibujar el árbol de búsqueda que realiza el proceso. Obtener únicamente la primera solución válida mostrando claramente el conjunto de conclusiones deducidas.

2. Explicar brevemente, apoyándose fundamentalmente en una resolución gráfica sobre árboles de búsqueda, cómo continuaría el proceso de resolución del problema para encontrar todas las soluciones válidas para el mismo caso del apartado anterior. No es necesario en este apartado explicar los detalles de cada paso realizado.

NOTA: Ante varias opciones durante el proceso de búsqueda, elegir primero las opciones que están antes según el orden que presentan en el enunciado.

EJERCICIO 6.4. Considerar un sistema mecánico S formado por tres clases de componentes A, B y C. El componente de tipo A puede ser A1, A2 o A3, el componente de tipo B puede ser B1, B2 o B3 y el componente C puede ser C1, C2 o C3. Además se utiliza la característica D con valores D1, D2 o D3 que indica las funciones posibles que puede ofrecer el sistema S. Para diseñar un sistema mecánico con dichos componentes se aplican los siguientes criterios:

1. Si A es A1 entonces B debe ser B2 o B3. Si A es A2 entonces B debe ser B1 o B2. Si A es A3 entonces B debe ser B1 o B3.
2. Si B es B1 entonces C debe ser C2 o C3. Si B es B2 entonces C debe ser C1 o C2. Si B es B3 entonces C debe ser C1 o C3.
3. Si C es C1 entonces A debe ser A2 o A3. Si C es C2 entonces A debe ser A2 o A3. Si C es C3 entonces A debe ser A1 o A3.
4. Si D es D1 entonces A debe ser A1. S D es D2 entonces A puede ser A1, A2 o A3. Si D es D3 entonces A debe ser A3.
5. Ante varias opciones posibles, es preferible la que tiene número de subíndice menor (por ejemplo, A2 es preferible a A3).
6. No son aceptables los diseños A1-B3-C3, A2-B1-C2, A3-B1-C2.

SE PIDE:

1. Formular los criterios anteriores para resolver este problema con el método de *configuración jerárquica HTN*. Mostrar los contenidos de las bases de conocimiento haciendo uso de una representación formal.

2. Aplicar el método de *configuración jerárquica HTN* para encontrar la primera solución sabiendo como dato que D es D2. Mostrar únicamente el árbol de resolución del problema.

3. Formular los criterios anteriores para resolver este problema con el método *proponer y revisar*. Mostrar los contenidos de las bases de conocimiento haciendo uso de una representación formal.

4. Aplicar el método *proponer y revisar* para encontrar la primera solución sabiendo como dato que D es D2. Mostrar únicamente el árbol de resolución del problema. Si tras proponer un máximo de 10 diseños no se encuentra la solución detener el proceso de búsqueda.

5. Comparar la aplicación del método *proponer y revisar* y del método *configuración jerárquica HTN* indicando cuál de ellos es más apropiado para ser utilizado en este problema de diseño mecánico.

7 Combinación de métodos

Los métodos de resolución de problemas revisados en los capítulos anteriores se pueden entender como piezas básicas para construcción de arquitecturas complejas de sistemas inteligentes. En el presente capítulo se describe la forma de construir arquitecturas haciendo uso de dichos componentes.

7.1. Diseño de arquitecturas compuestas

El conjunto de métodos de resolución de problemas identificados supone una recopilación y estandarización de formas de razonar para resolver problemas observadas en personas, lo cual es una herramienta muy útil como ayuda a la construcción de sistemas complejos. La identificación de dichos métodos puede verse como una forma de reutilizar conocimiento de resolución de problemas. Tal como se muestra en la figura 7.1 el conocimiento puede considerarse dividido en dos tipos: (1) de resolución de problemas y (2) del dominio. El primero puede expresarse de forma general para realizar tareas tales como diagnosticar, planificar, configurar, etc. El segundo se refiere al conocimiento sobre el dominio en donde se lleva a cabo la resolución de problemas que pueden ser por ejemplo áreas profesionales tales como medicina, electrónica, mecánica, economía, etc.

Aunque normalmente existe una estrecha relación entre la forma de razonar sobre un dominio y la forma de representarlo, sin embargo es interesante plantear la separación anterior con el fin de mostrar posibilidades de reutilización. Por ejemplo, es útil describir conocimiento sobre diagnóstico de forma separada del dominio donde se lleva a cabo, con objeto de reutilizar dicho conocimiento a través de diferentes dominios (mecánica, medicina, etc.). Esta es precisamente la idea que está detrás de los métodos de resolución de problemas descritos en los capítulos previos. De forma dual, también es posible pensar en reutilización de dominios, como es el caso de mecánica, para diferentes tareas (configuración, diagnóstico, etc.). Para ello el conocimiento del dominio debe representarse (al menos parcialmente) de forma declarativa sin prejuzgar el uso que se vaya a hacer. Detrás de esta idea de reutilización está el concepto de *ontología* utilizado recientemente en Ingeniería del Conocimiento con el fin de potenciar las posibilidades de reutilización.

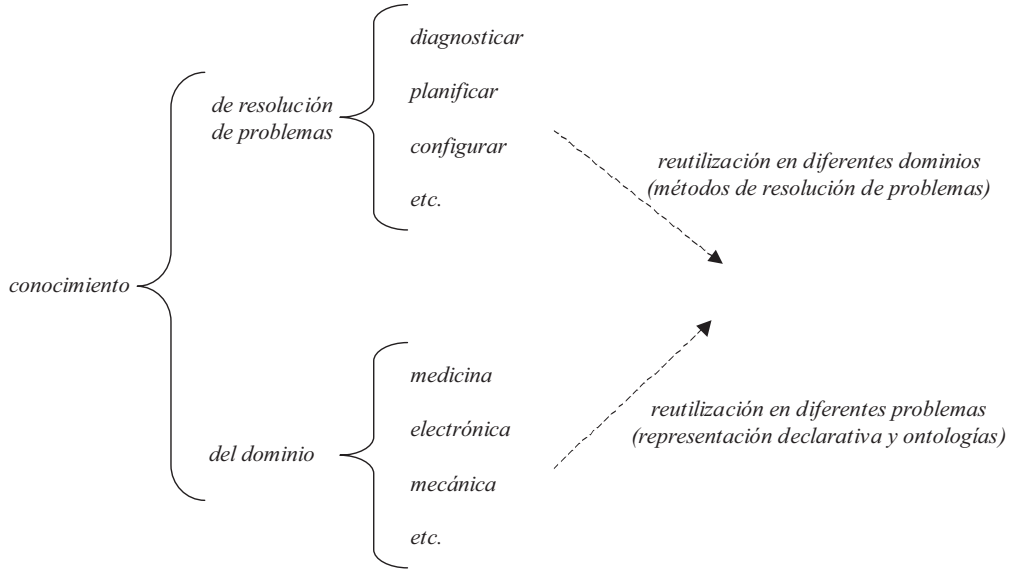


Figura 7.1: Opciones de reutilización de conocimiento.

La construcción de un sistema complejo con ayuda de los métodos se muestra en la figura 7.2. El desarrollador dispone de una biblioteca de métodos de resolución de problemas (clasificación jerárquica, cubrir y diferenciar, etc.). Cuando se enfrenta al

análisis de un problema con el objetivo de construir un sistema informático orientado a su resolución de forma automática, en primer lugar, estudia las características de dicho problema observando cómo es resuelto de forma manual por personas especializadas. El hecho de que la biblioteca de métodos suponga una recopilación de formas típicas de resolver problemas indica que existe una alta probabilidad de que alguno o algunos métodos coincidan con la forma en que las personas resuelven manualmente el problema bajo estudio dando como lugar la selección de un método para su resolución total o parcial.

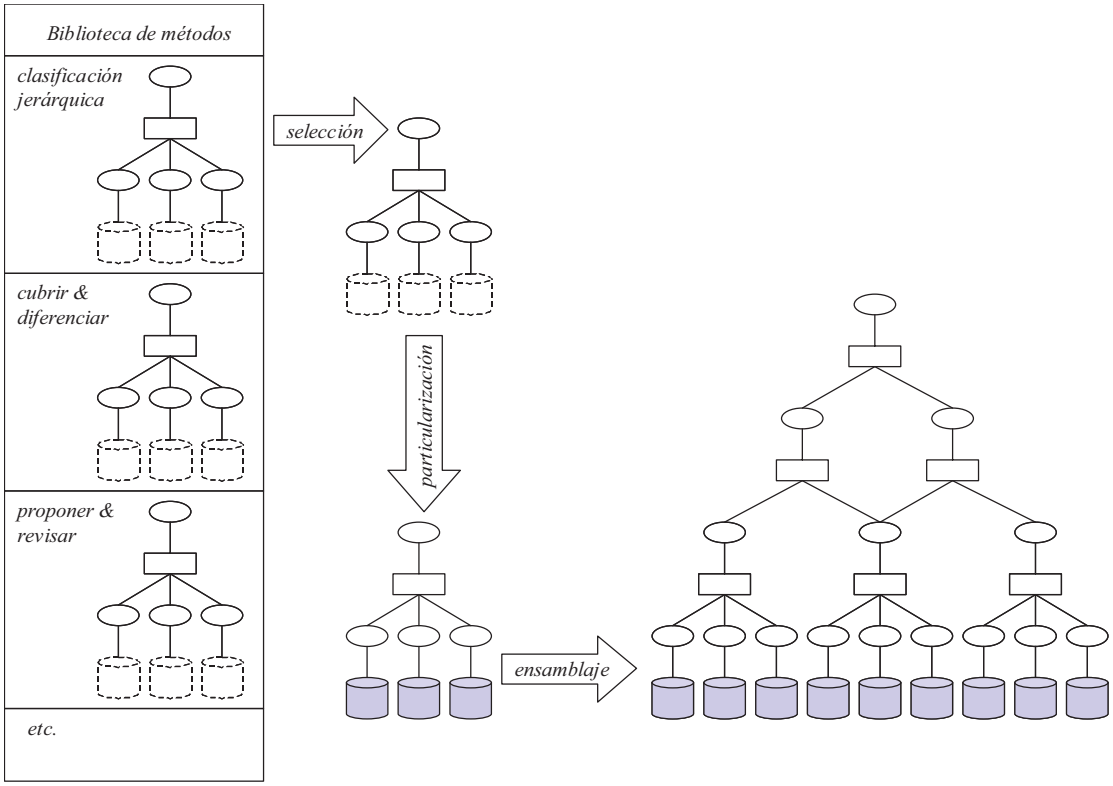


Figura 7.2: Visión general del proceso de construcción de un sistema con ayuda de métodos.

Dado que el método se encuentra definido de forma abstracta para ser aplicable a un conjunto amplio de problemas concretos, el siguiente paso es llevar a cabo una particularización del mismo en el dominio en donde se trate el problema. En este

caso, el método sirve de guía para identificar las diferentes clases de conocimiento que deben ser obtenidas de los expertos en el dominio. Como resultado de este proceso, se obtiene una estructura del método en donde las bases de conocimiento están formadas por los diversos elementos específicos del dominio. El conocimiento de control del método puede también ser objeto de adaptaciones y refino de acuerdo con las particularidades del problema concreto.

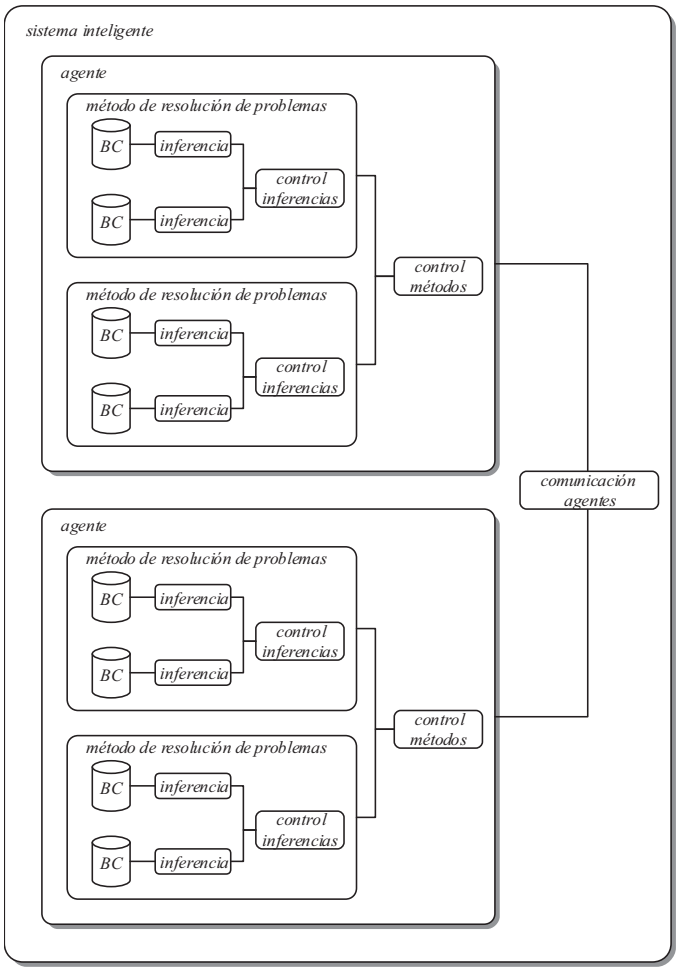


Figura 7.3: Componentes de un sistema inteligente en distintos niveles de agregación.

Finalmente, el método seleccionado es ensamblado en la arquitectura completa junto con otros métodos. Dicho ensamblaje se puede visualizar como una jerarquía

de tareas y métodos que en la base está soportada por un conjunto de bases de conocimiento. La tarea o tareas de más alto nivel se van descomponiendo en tareas más sencillas (elipses en la figura) con ayuda de métodos (rectángulos en la figura).

El proceso de construcción descrito presenta ciertos aspectos comunes a diversos enfoques de diseño en otros campos. En primer lugar, este planteamiento sigue el enfoque de construcción de sistemas *basada en componentes*. Bajo esta perspectiva, un sistema se construye mediante el ensamblaje de componentes que han sido previamente adaptados (tal como se realiza por ejemplo para construir un sistema digital con diversos componentes electrónicos). En este caso, los componentes son los métodos de resolución de problemas que se adaptan mediante refino y particularización en el problema. Por otro lado, se puede mencionar también la idea de *patrones de diseño* que se presenta en el área de ingeniería del software. Este concepto persigue un objetivo similar al de los métodos de resolución de problemas dado que identifica plantillas de diseño de tipos de sistemas que sirven de guía en el proceso de construcción de sistemas concretos. No obstante, la idea de patrones de diseño está definida a un nivel más cercano a la programación bajo un enfoque de proceso de datos, a diferencia de los métodos de resolución de problemas que se plantean como una forma de organizar y procesar conocimiento.

La figura 7.3 muestra una visión global de los diferentes componentes de un sistema inteligente en distintos niveles de agregación. A un primer nivel, de forma general un sistema inteligente se puede entender formado por un conjunto de agentes de forma cooperativa contribuyen a la resolución del problema. La coordinación entre dichos agentes se entiende distribuida entre ellos (no centralizada) siguiendo algún modelo de los que se manejan en los enfoques de sistemas multiagente. Un agente se puede entender formado por un número de métodos de resolución de problemas junto al conveniente conocimiento de control para su manejo durante el proceso de resolución. Cada método, a su vez, estará formado por varios pares <base de conocimiento, inferencia> junto al conocimiento de control para manejo de

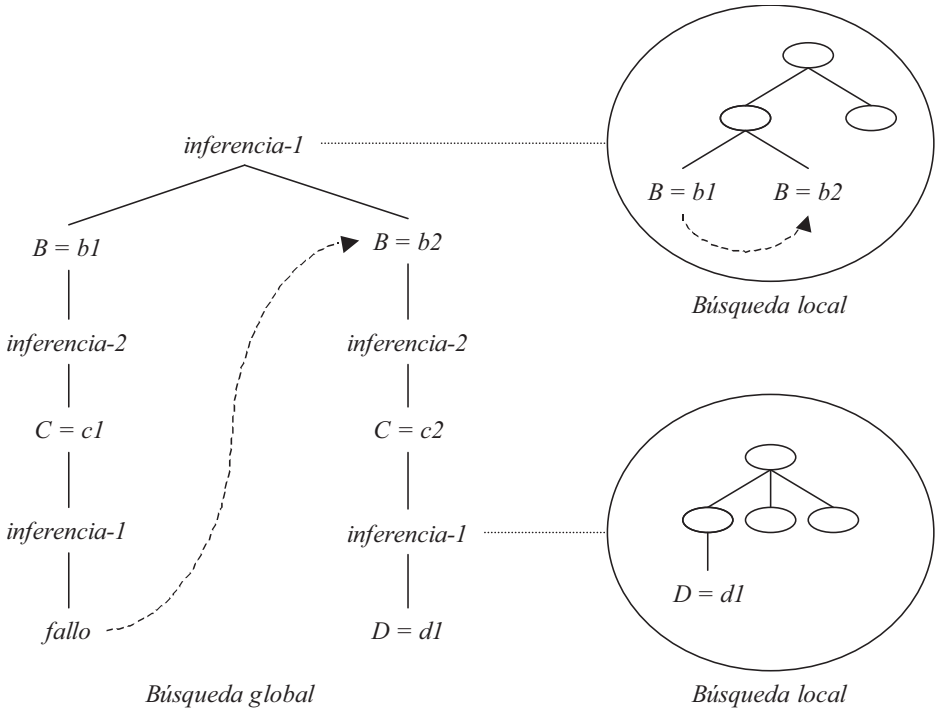
inferencias. Finalmente, cada base de conocimiento estará formada por los elementos de representación que correspondan en cada caso (reglas, marcos, etc.).

Los elementos correspondientes al control de métodos, control de inferencias y las inferencias normalmente se suelen representar mediante enfoques procedimentales que expresan la forma de conectar entradas y salidas junto al orden en que se ejecutan los diversos pasos (régimen de control). Sin embargo, se pueden presentar situaciones complejas para combinar búsquedas locales como la que muestra el siguiente ejemplo que pueden requerir soluciones de representación alternativas. Se tienen 2 pasos de inferencia: `inferencia-1(x -> y)` e `inferencia-2(x -> y)`. La forma de conectar entradas y salidas, y el orden de ejecución es el siguiente:

```
1. inferencia-1(A -> B)
2. inferencia-2(B -> C)
3. inferencia-1(C -> D)
```

Cuando comienza el proceso, inferencia-1 realiza una búsqueda local y genera una primera salida *B=b1* como solución provisional. Localmente, inferencia-1 ha desarrollado un árbol de búsqueda que le ha permitido llegar a *B=b1* aunque, en caso de que sea requerido, podría continuar el proceso de búsqueda generando otras opciones diferentes para *B*, por ejemplo *B=b2*. Esto queda recogido en el árbol local construido por inferencia-1 cuyo desarrollo queda congelado en ese momento. A continuación se ejecuta inferencia-2 que genera la salida *C=c1*. En el paso 3, cuando se invoca de nuevo a inferencia-1, es importante hacer notar que se inicia un nuevo proceso de búsqueda diferente al realizado en el paso 1. Para ello, es necesario almacenar el árbol desarrollado en paso 1 por si fallos posteriores requieren volver para generar una salida alternativa, tal como muestra la figura 7.4.

Esta forma de proceso, si se programa mediante enfoques procedimentales tradicionales, requiere que el programador deba gestionar de forma explícita pilas de estructuras de datos para permitir el anidamiento indefinido de llamadas (similar al proceso que se realiza en programación recursiva, aunque en ese caso sin intervención del programador), junto a modos de ejecución y estados de control de pasos de inferencia, lo que puede dar lugar a programas heterogéneos más oscuros y más difíciles de mantener. Como alternativa pueden utilizarse enfoques como los que manejan lenguajes de programación con proceso no lineal como es el caso del modelo de computación del lenguaje Prolog.



7.4: Ejemplo de combinación de inferencias con búsquedas locales.

Por otra parte, los procesos de búsqueda locales llegan a conjuntos de conclusiones que más adelante pueden necesitar ser revisadas dentro de un proceso de búsqueda global para plantear caminos alternativos. En general, es deseable ser capaz de distinguir entre conclusiones que deben cambiarse y conclusiones que se mantienen

con el fin de aprovechar el esfuerzo realizado en la deducción de hechos válidos. Esto se da, por ejemplo, en el método de proponer y revisar que debe realizar cambios en diseños y en donde es útil evitar recalcular aspectos del diseño que no van a cambiar. Por otra parte, en las búsquedas es interesante recoger las explicaciones que justifican el fallo en opciones previamente planteadas sin éxito con el fin de reducir las posibilidades de iniciar caminos de búsqueda erróneos. Estas razones son la base del uso de los denominados TMS (*Truth Maintenance Systems*) o sistemas de mantenimiento de la verdad.

En resumen, la forma de articular los diferentes componentes que realizan búsquedas locales dentro de un proceso de búsqueda global puede requerir el uso de modelos de control más sofisticados que los que se manejan en lenguajes de programación convencional, como son los mecanismos de computación no lineal (tipo Prolog), sistemas de mantenimiento de la verdad TMS, y otros tales como el modelo de agendas o el modelo de pizarra.

7.2. Ejemplos de sistemas

La tecnología basada en el conocimiento se ha venido aplicando desde sus comienzos en los años 70 en diversos problemas. En particular en problemas de monitorización, diagnóstico de fallos, ayuda al diseño y, más recientemente, ayuda a la toma de decisiones en entornos complejos. En esta sección se describen ejemplos de sistemas de esta clase en donde la utilización de los métodos de resolución de problemas ha permitido aportar soluciones efectivas.

7.2.1 Gestión de emergencias por inundaciones

En esta sección se presenta el caso del sistema SAIDA para ayuda a la decisión en situación de emergencias [Cuena, Molina, 99; Molina, Blasco, 03]. Este sistema se ha desarrollado encuadrado en un programa nacional (Programa SAIH, Sistema Automático de Información Hidrológica) destinado a instalar una infraestructura de información avanzada en las cuencas hidrográficas. Esta iniciativa, promovida por el Ministerio de Medio Ambiente en España, que se está llevando a cabo desde los años ochenta, incluye el uso de tecnologías avanzadas para sensorización del estado de los ríos (lluvia, niveles y caudales en los ríos, niveles en embalses, etc.).

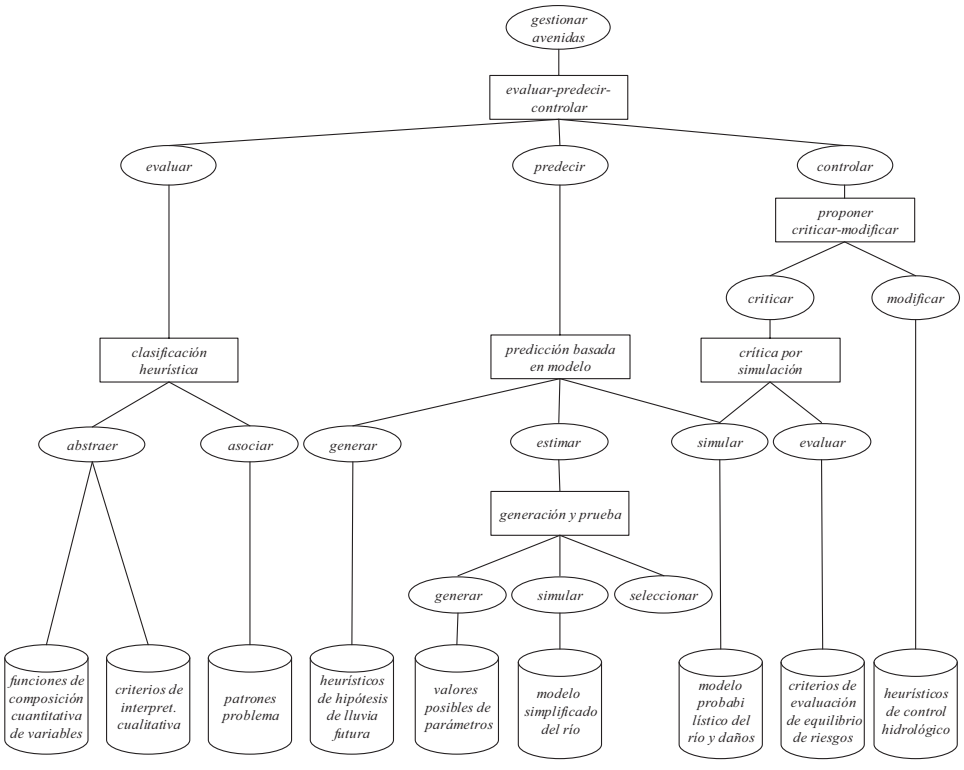


Figura 7.5: Estructura de tareas, métodos y bases de conocimiento de SAIDA

Uno de los principales propósitos de la instalación de esta infraestructura es ayudar a los responsables en la toma de decisiones ante la presencia de inundaciones por desbordamiento de los ríos. El sistema SAIDA (Sociedad de Agentes Inteligentes para Decisión en Avenidas) se ha construido con el fin de asistir en la toma de decisiones en situaciones de emergencias producidas por inundaciones, utilizando como datos de entrada las medidas registradas por el sistema de información SAIH. SAIDA está concebido como un asistente que interacciona con el operador mediante un diálogo dirigido a suministrar la suficiente información para que el operador pueda asumir la responsabilidad de las decisiones sobre las acciones de control a realizar ante una situación de emergencia.

SAIDA está en línea con el concepto de sistema informático denominado *intelligent assistant* [Boy, Gruber, 90] que plantea la disponibilidad de sistemas informáticos en contextos donde la tarea a realizar debe entenderse como una actividad cooperativa entre el hombre y la máquina de forma tal que parte de los procesos pueden ser llevados a cabo automáticamente por el ordenador, pero otros han de ser asumidos necesariamente por la persona. La idea de asistente inteligente enfatiza una forma de comunicación entre ordenador y usuario en donde (1) se maneja un medio de comunicación (lenguaje, gráficos, etc) a un adecuado nivel de abstracción, eficaz para las tareas a realizar, y (2) el sistema simula conciencia respecto a la forma en que automáticamente resuelve problemas con el fin de ser capaz de justificar sus conclusiones mostrando el conocimiento empleado de una forma análoga a como lo realizan los responsables de las decisiones.

Para cumplir este objetivo SAIDA ofrece al operador un repertorio de tipos de consultas que pueden agruparse alrededor de tres fases sobre los diversos puntos de atención, de forma que se consideran tres clases de preguntas que el sistema SAIDA responde para dar soporte a la decisión:

- *Evaluación* de la situación actual, con el tipo de pregunta ¿qué está pasando? para evaluar la gravedad de la situación hidrológica actual.

- *Predicción* a corto plazo, con el tipo de pregunta ¿qué puede pasar?, con objeto de analizar el comportamiento previsible considerando diversas hipótesis de evoluciones posibles a corto plazo.

- *Recomendación* de acciones de control, con el tipo de pregunta ¿qué hacer?, para determinar acciones de control a realizar que reduzcan los posibles daños, admitiendo también diversas hipótesis de comportamiento futuro.

Estas tres fases de desarrollan en un entorno con incertidumbre. Existe incertidumbre en la función de medida, en cuanto que las variables de estado relevantes desde el punto de vista hidrológico no son directamente observables mediante la instrumentación disponible. Por ejemplo, la lluvia real no se conoce con precisión, dado que los pluviómetros están distribuidos espacialmente con distancias significativas. Existe también incertidumbre en el pronóstico de evolución futura, en cuanto que las técnicas de simulación con modelos matemáticos numéricos de la dinámica de la cuenca hidrológica son simplificaciones de la realidad, que utilizan parámetros que no se conocen suficientemente. Por ejemplo, los parámetros que indican la capacidad de infiltración del terreno son difíciles de calibrar lo que hace que, normalmente, se aproximen en tiempo real en función de los datos medidos recientemente. Finalmente, la propia decisión es incierta, en cuanto que se desconocen los efectos reales que tendrán las medidas de control o el grado de respuesta de la población afectada. El manejo de estas múltiples fuentes de incertidumbre presenta dificultades que hacen adecuado el uso de técnicas de representación simbólica y manejo de métodos de tratamiento de incertidumbre.

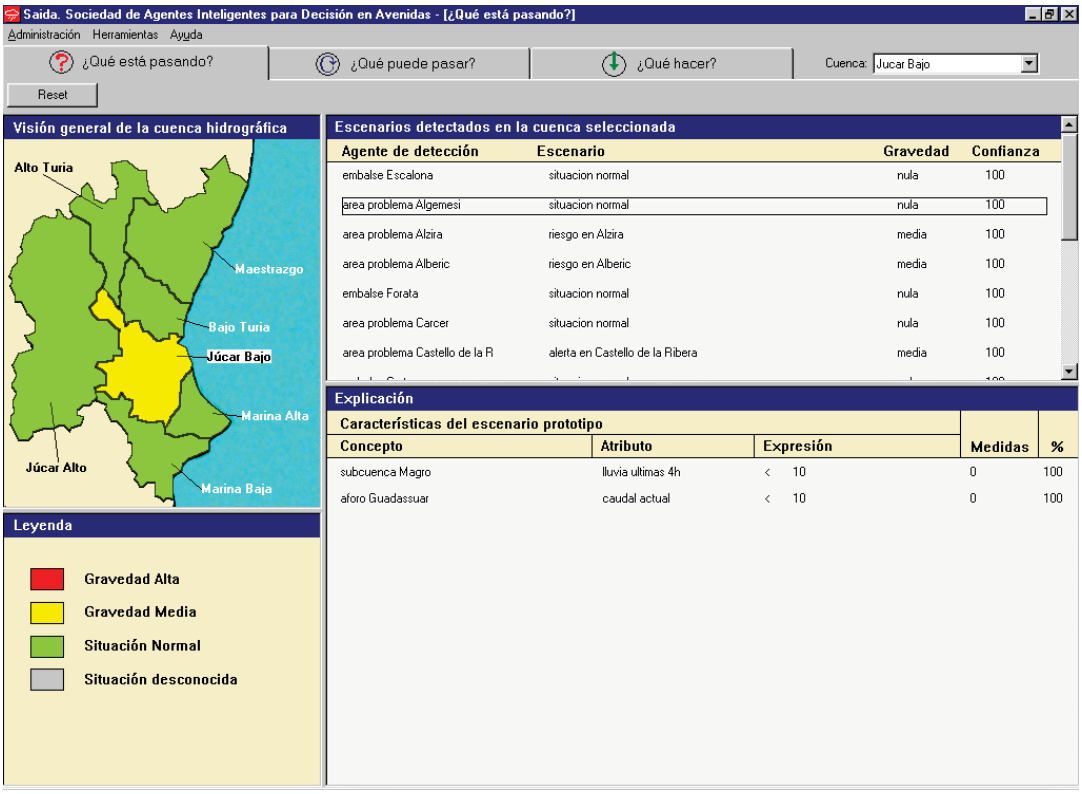


Figura 7.6: Ejemplo de presentación la evaluación de la situación actual.

Para realizar la tarea de evaluación de la situación actual se utilizó una adaptación del método de clasificación heurística que se basa en el manejo de bases de conocimiento que incorporan criterios empíricos para la clasificación de información de entrada. La evaluación de la situación actual recibe como entrada las medidas de pluviómetros, aforos y sensores de nivel que forman el sistema de información SAIH y, a continuación, lleva a cabo un proceso basado en el conocimiento que comprende dos fases principales:

- *Abstracción de los datos*. Esta fase incluye por un lado el preproceso de medidas, en la que, a partir de las series temporales medidas por sensores, se calculan otras medidas necesarias en la evaluación de la situación actual. Para llevar a cabo este proceso se cuenta con un modelo que se formula con un lenguaje funcional que permite describir expresiones aritméticas, estadísticas,

etc. para relacionar las variables. Por otro lado, se realiza también una interpretación cualitativa de valores de variables, en la que se obtiene el valor cualitativo correspondiente a determinadas medidas numéricas. El modelo para llevar a cabo dicha interpretación se formula mediante una representación en reglas en forma de sentencias condicionales del tipo SI <premisas> ENTONCES <conclusión>.

```
PATRON emergencia en Sumacarcer
DESCRIPCION
  (subcuenca Naranjero)
          lluvia ultimas 24h      < 200                 [a],
  (subcuenca Naranjero)
          lluvia ultimas 4h       > 80                  [b],
  (embalse Naranjero)
          volumen actual          > 20                  [c],
  (subcuenca Tous)
          lluvia ultimas 24h      < 200                 [d],
  (subcuenca Tous)
          lluvia ultimas 4h       > 80                  [e],
  (embalse Escalona)
          volumen actual          > 5                   [f],
  (embalse Tous)
          volumen actual          > 100                 [g],
  (area problema Sumacarcer) gravedad = alta,
RELEVANCIA DE CARACTERISTICAS
   (a;b),c,d,e,f,g -> 100%.
   a, b, c -> 60%.

PATRON alerta en Sumacarcer debido a lluvias en Tous
DESCRIPCION
  (embalse Tous)
          volumen actual          pertenece [120,130]   [a],
  (embalse Escalona)
          volumen actual          pertenece [4,7]       [b],
  (subcuenca Tous)
          lluvia ultimas 4h       > 80                  [c],
  (subcuenca Tous)
          lluvia ultimas 24h      < 200                 [d],
  (subcuenca Naranjero)
          lluvia ultimas 4h       < 80                  [e],
  (area problema Sumacarcer) gravedad = alta,
RELEVANCIA DE CARACTERISTICAS
   a,b,c,d,e -> 100%.
...
```

Figura 7.7: Ejemplos de patrones para representar escenarios de alerta.

- *Asociación* mediante equiparación con escenarios de alerta. Para ello, dada una situación descrita por los datos de sensores y el resultado de la abstracción, si ésta equipara con algún escenario de alerta prefijado, se obtiene un nivel de gravedad. Los escenarios de alerta se formulan en forma de marcos, es decir, patrones de situaciones definidos mediante sus características. La figura 7.7 muestra ejemplos de patrones de este tipo.

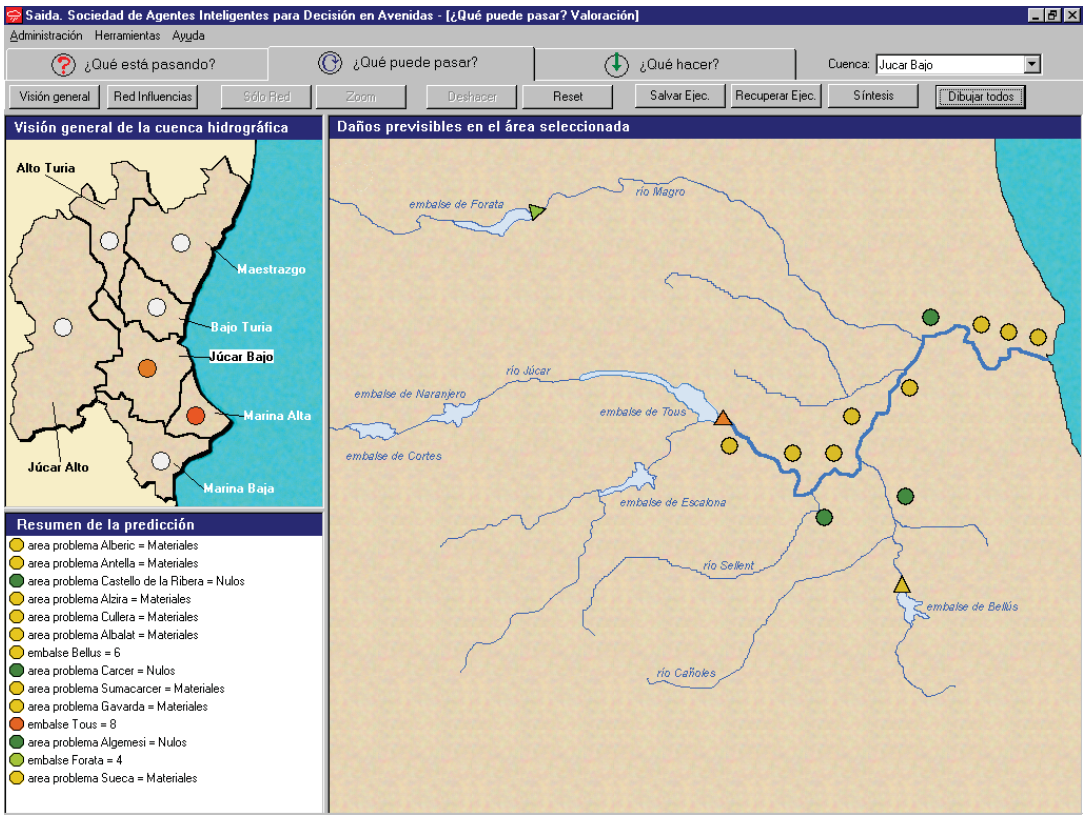


Figura 7.8: Pantalla que muestra los resultados generales de la predicción.

La función de predicción tiene como objetivo mostrar cuál puede ser el comportamiento global de la cuenca y los futuros daños que se pueden producir. SAIDA presenta los resultados de la predicción en una ventana general (figura 7.8) identificando las zonas con potenciales problemas (áreas problema o embalses).

Para realizar la predicción se utiliza un método de predicción basada en el uso de un modelo del comportamiento del río. Este método lleva a cabo tres etapas: (1) generar una hipótesis de lluvia futura en la cuenca, (2) estimar cuál es el estado actual que servirá como estado inicial de la simulación, (3) simular el comportamiento haciendo uso del modelo del río y de los daños. Para realizar la hipótesis de cuál será la lluvia futura en la cuenca se parte de la predicción general proporcionada por el Instituto Nacional de Meteorología, una información muy general en toda la zona para las próximas 6, 12 y 24 horas. La hipótesis realiza un

refino de dicha predicción de lluvia, distribuyendo en horas y zonas, basándose en criterios heurísticos sobre forma habitual de llover en dicha zona y posicionamiento en situaciones más pesimistas.

La estimación del estado inicial es necesaria para asumir los valores de ciertos parámetros Ci (denominados *números de curva*) relacionados con la infiltración del terreno que no son medibles directamente por sensores. Para ello se realiza una estimación mediante un proceso de generación y prueba que, primero genera hipótesis de valores para el parámetro Ci siguiendo un orden prefijado (para cada parámetro hay un total de 10 opciones posibles y en un modelo hay aproximadamente unos 10 parámetros Ci). A continuación se realiza una simulación simplificada haciendo uso de un modelo sencillo numérico que genera una respuesta de caudal Qs (caudal simulado) para cada valor de parámetro. El valor de parámetro que tenga una caudal Qs más cercano al medido por sensores Qm será el valor elegido para dicho parámetro.

Para realizar la simulación del comportamiento se utiliza un modelo del comportamiento del río y los daños que se pueden generar. Con el fin de tener en cuenta los aspectos de incertidumbre en el problema, el sistema SAIDA utiliza redes bayesianas para formulación del modelo de comportamiento. La estructura de cada río se descompone en los diversos fenómenos físicos a considerar (lluvia, infiltración, transporte, embalse, etc.) formando una red global de procesos en donde cada uno se lleva a cabo mediante una red local bayesiana. La red global forma una estructura de influencias que representa el encadenamiento lógico de los diferentes fenómenos de acuerdo con la topología de la cuenca. Por ejemplo, la figura 7.9 muestra un ejemplo pantalla que muestra el sistema SAIDA para justificar una determinada predicción. En dicha pantalla se muestra la red de influencias correspondiente a una determinada cuenca en donde el operador puede consultar la predicción de cualquier variable e incluso puede modificar el valor de alguna para realizar un análisis del tipo *qué-pasaría-si*. Por ejemplo, los datos

relativos a lluvias (tanto medidos como hipótesis de futuro) pueden ser modificados por el usuario con el fin de poder realizar distintas hipótesis más pesimistas u optimistas. También, además de otras variables, puede modificarse la estrategia de operación en los embalses para comprobar el efecto posible. Por tanto, el uso de dicha estructura es útil además de como forma de justificar predicciones, como herramienta para el análisis de las diferentes respuestas de la cuenca ante diferentes hipótesis de lluvia o gestión de embalses planteadas por los responsables de las decisiones.

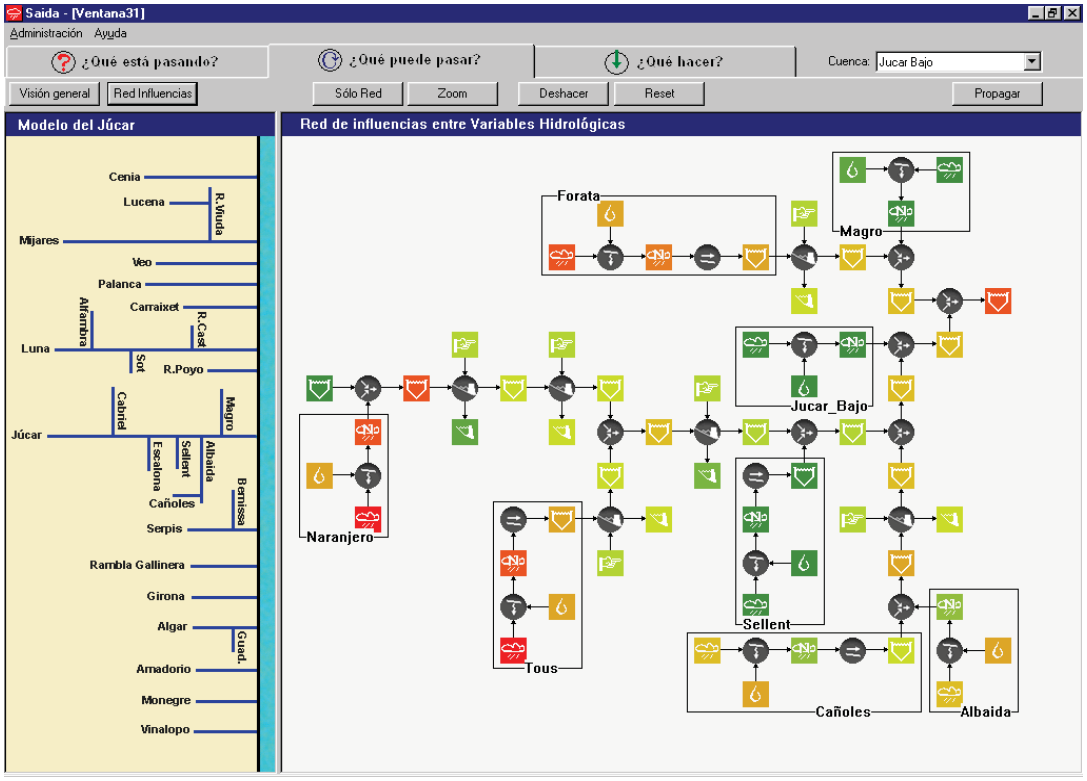


Figura 7.9: Pantalla que muestra la cadena lógica para predicción de comportamiento.

El modelo de redes bayesianas opera con distribuciones de probabilidad asociados a valores cualitativos, de forma que normalmente se tiene un rango de valores con la probabilidad de cada uno de ellos para cada variable cualitativa. De esta forma se puede ofrecer al ingeniero responsable de las decisiones una visión sintética

comprensiva de la situación esperable a corto plazo en la red hidrográfica. Para cada variable SAIDA indica la posible evolución para las próximas horas (por ejemplo, las próximas 8 horas) en una escala cualitativa y afectada por medidas de certeza expresadas en forma de distribuciones de probabilidad. La figura 7.10 muestra un ejemplo de presentación de los valores de una variable. La variable lluvia neta en Naranjero tiene un conjunto discreto de posibles valores (eje vertical) que toman valores a lo largo de diferentes instantes consecutivos (eje horizontal). En cada instante se tiene una distribución de valores de probabilidad asociada a los valores posibles de la variable. En la presentación, para cada instante, se identifican los centros de gravedad de cada distribución que se muestran unidos por una línea para resaltar la tendencia.

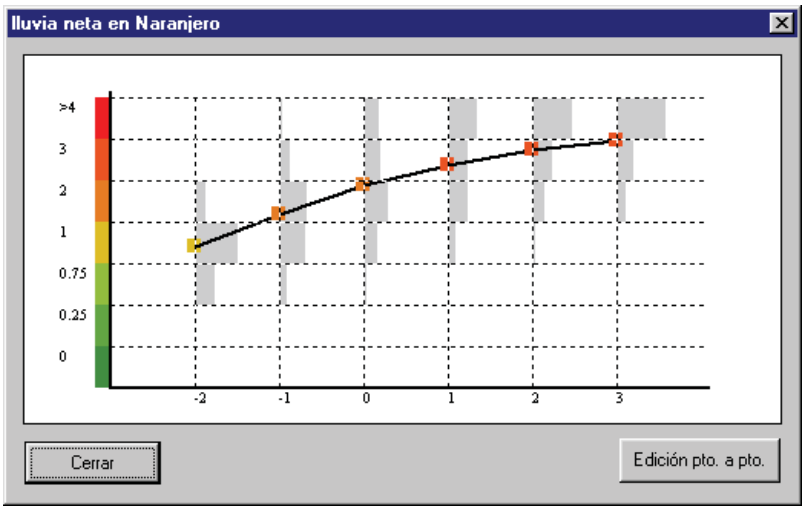


Figura 7.10: Predicción del comportamiento de una variable hidrológica.

Uno de los objetivos principales de SAIDA es ofrecer un conjunto de recomendaciones de actuación en los elementos de control del sistema (embalses) que permitan reducir los efectos de las avenidas. De igual forma, SAIDA puede proporcionar una serie de recomendaciones de actuación dirigidas a los

responsables de protección que minimicen los impactos en los núcleos de población, redes de transporte, etc., que no se hayan podido evitar mediante una actuación en los embalses. La figura 7.11 muestra un ejemplo de pantalla que presenta SAIDA sobre actuaciones globales en embalses, junto con la línea argumental (mostrada tanto en texto como gráficamente) que justifica dicha actuación.

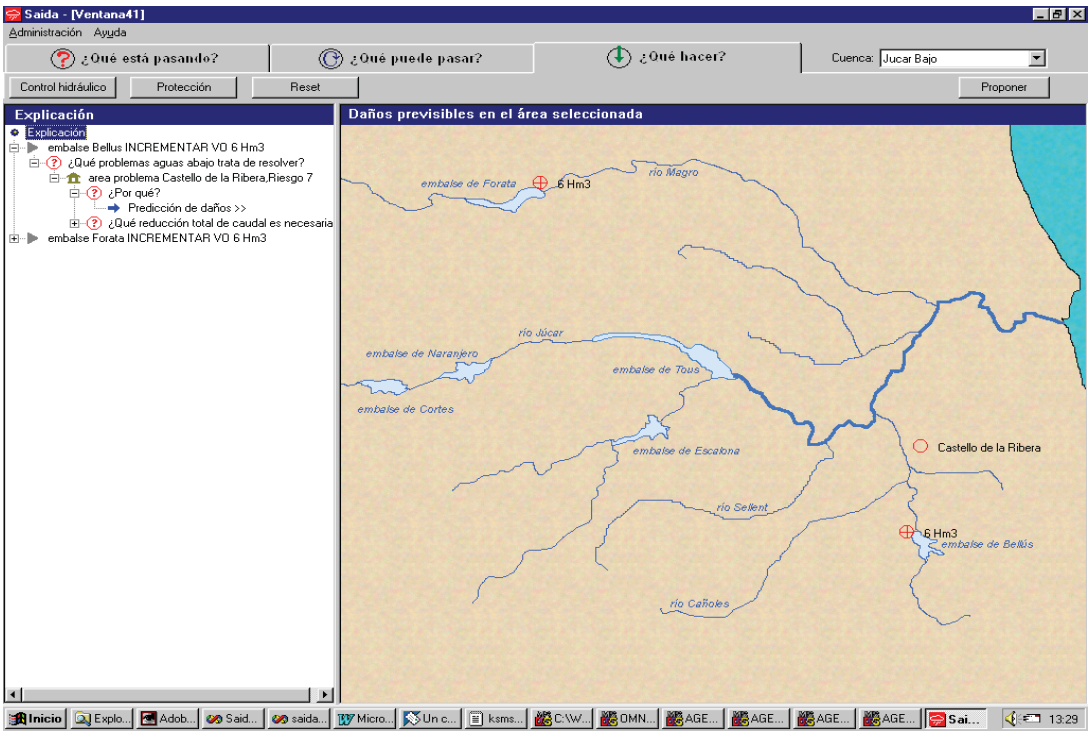


Figura 7.11: Pantalla de SAIDA mostrando recomendaciones de actuación.

Las recomendaciones de actuación propuestas para los diferentes embalses se definen como *volúmenes objetivo*, los cuales representan el volumen al que cada embalse debe llegar en un horizonte de tiempo determinado. En base a los volúmenes objetivo recomendados los operadores de los embalses programarán los

órganos de desagüe de la forma más adecuada para conseguir alcanzar dichos volúmenes. La propuesta de control de SAIDA es una asignación de volúmenes objetivos a cada embalse. El espacio de posibles soluciones es el conjunto total de todas las asignaciones posibles a cada embalse. La forma de determinar una solución es mediante un proceso inspirado en la estrategia proponer-criticar-modificar para problemas de configuración.

Para determinar las acciones de control en embalses se realiza una búsqueda dirigida a equilibrar el riesgo asumido en áreas inundables y embalses. Para ello se maneja un concepto global, que representa dicho riesgo en una escala discreta de 0 a 10 y cuya evaluación se realiza individualmente en cada zona del río de acuerdo con un criterios locales de la situación esperable en la zona. Esta fase se realiza distribuida mediante agentes (agentes de áreas inundables y agentes de embalses). Para proponer acciones sobre embalses SAIDA aplica una variante distribuida del método general para problemas de configuración *proponer-criticar-modificar* que realiza los siguientes pasos::

1. *Crítica (inicial).* partiendo del estado actual del río y de la predicción, se evalúa el desequilibrio de riesgos en cada punto de la cuenca (áreas inundables y embalses). Si dicho desequilibrio es significativo se realiza el paso (2).

2. *Modificación*. Se generan hipótesis de actuación en embalses basadas en heurísticos de control de embalses orientadas a reducir los desequilibrios (por ejemplo, si el riesgo de una zona inundable es muy alto y el riesgo en un embalse aguas arriba es bajo, dicho embalse puede incrementar su volumen objetivo). Como resultado se genera una propuesta de control formada por un conjunto de actuaciones en embalses.

3. *Crítica*. La propuesta de control se simula con el modelo de comportamiento de río y, a continuación, se evalúa el desequilibrio de riesgos con el fin de comprobar si se ha reducido. Si no es así se vuelve al paso (2) para continuar la búsqueda..

Para formular los heurísticos de control hidráulico se utiliza una representación formal basada en reglas de tipo *SI <premisas> ENTONCES <conclusión>* junto con una representación lógica (cláusulas de Horn). Respecto a las estrategias de defensa para protección ante avenidas (aspecto no mostrado en la figura de tareas, métodos y bases de conocimiento), SAIDA utiliza modelos locales formulados en reglas que relacionan situaciones de inundación con las diferentes medidas de protección.

El diseño final del sistema SAIDA sigue una filosofía multiagente, que facilita la organización del conocimiento que se maneja en la aplicación. Este tipo de organización es la que da nombre al sistema SAIDA: *Sociedad de Agentes Inteligentes para Decisión en Avenidas*. Los tipos de agentes que forman el sistema se han distribuido de acuerdo con (1) una descomposición funcional, siguiendo las funciones principales que se observan en el sistema, lo que ha dado lugar a cuatro tipos de agentes y (2) una descomposición geográfica, en donde se obtienen cada uno de los agentes concretos según su localización geográfica. Los cuatro tipos de agentes considerados son los siguientes:

- *Agente de área problema*. Este tipo de agente corresponde a una parte especializada en los daños posibles que se pueden dar en un área inundable. El agente área problema incluye conocimiento sobre una zona geográfica de la cuenca en donde pueden presentarse problemas de inundación. Normalmente se definirán agentes de área problema en zonas relacionadas con núcleos de población o terrenos que presenten una problemática particular.

- *Agente de comportamiento hidrológico*. Este tipo de agente está especializado en la simulación del comportamiento de una cuenca. El agente de comportamiento hidrológico incluye conocimiento a un cierto nivel de abstracción sobre los fenómenos físicos relacionados con la propagación de agua por los cauces de la cuenca. Habitualmente, habrá un agente de comportamiento para cada río de la cuenca hidrográfica.

- *Agente de gestión de embalses*. Este tipo de agente está especializado en estrategias de gestión de embalse para producir un determinado desagüe a la salida. El agente de gestión de embalse incluye conocimiento sobre las decisiones estratégicas relativas al manejo de un embalse para operar ante situaciones problemáticas debidas la presencia de avenidas. Se definirá un agente de este tipo para cada embalse. Este agente no incluye conocimiento sobre simulación cuantitativa del embalse, sino que dicha información está englobada en el agente de comportamiento.

- *Agente de protección*. Este tipo de agente modeliza el conocimiento de gestión de recursos orientados a la protección frente a la presencia de inundaciones. El agente de gestión de recursos es responsable de las decisiones sobre manejo de recursos de protección contra las inundaciones. Por ejemplo recursos relacionados con funciones hospitalarias (ambulancias, camas de hospitales, etc.), equipos de actuación (bomberos, protección civil, ejército, guardia civil, etc.), etc. Se definirán tantos agentes como grupos de recursos haya que manejar en las decisiones sobre protección.

El modelo de conocimiento de cada uno de los agentes, se divide en dos niveles: un nivel local o de resolución de problemas y un nivel social. El nivel de resolución de problemas recoge las competencias locales del agente en cuanto a conocimiento del

dominio y métodos de resolución de problemas. El nivel social es un componente metanivel con el que se simula la capacidad del agente de razonar sobre sus propias competencias y sobre las de los demás agentes.

Un aspecto importante a tener en cuenta para utilizar con éxito sistemas inteligentes de ayuda a la decisión es el esfuerzo de construcción y mantenimiento de los modelos propios de cada cuenca que recogen el conocimiento sobre comportamiento y actuación específico en las zonas concretas. Para facilitar la construcción de dichos modelos es útil disponer de herramientas software especiales que ofrezcan ayudas de edición, mantenimiento de coherencia, utilidades gráficas, etc. Dichas herramientas pueden ofrecer imágenes y lenguajes más cercanos a los profesionales de hidrología además de incluir utilidades adicionales de aprendizaje automático. En este sentido, SAIDA maneja la herramienta CAM-HIDRO [Molina, Blasco, 03] un entorno software que permite el manejo de complejos modelos con diferentes imágenes gráficas acerca de las diversas bases de conocimiento junto con las ayudas a edición y coherencia mencionadas.

SAIDA ha sido desarrollado con ayuda de la herramienta KSM dando lugar a una arquitectura distribuida en donde operan en paralelo varios procesos KSM (entre cuatro y ocho procesos) en donde cada uno da soporte a una familia de agentes de la misma clase. Dichos procesos se conectan mediante un mecanismo software que emula la interacción social y se comunican correspondiente interfaz gráfico de usuario. Con ello, KSM da soporte a cuatro estructuras generales de conocimiento, lo que supone un total de 33 tipos de bases de conocimiento. El modelo concreto realizado para la cuenca del río Júcar da lugar a un modelo de 26 agentes. De dicho conjunto, 7 agentes se encargan de la gestión de embalses lo que hace un total de 41 bases de conocimiento particulares. Estas dimensiones muestran el interés de utilizar sistemas avanzados de ingeniería del conocimiento para compartir y reutilizar conocimiento con objeto de reducir el esfuerzo de desarrollo y facilitar la validación y mantenimiento del sistema final.

SAIDA fue inicialmente basado en trabajos de I+D previos desarrollados en el ámbito universitario tales como CYRAH [Cuena et al., 91], [Cuena et al. 92], que reúne métodos de razonamiento simbólico con métodos de simulación numérica y el sistema SIRAH [Alonso et al., 90] que modela el conocimiento sobre comportamiento hidrológico haciendo uso de formulaciones cualitativas debidamente manejadas con criterios de control. SAIDA supone una maduración y actualización de dichas experiencias. SAIDA ha sido aplicado en las cuencas del río Júcar y del Sur (ríos Guadalhorce y Guadalmedina), mostrando la utilidad de las soluciones técnicas aportadas por el sistema.

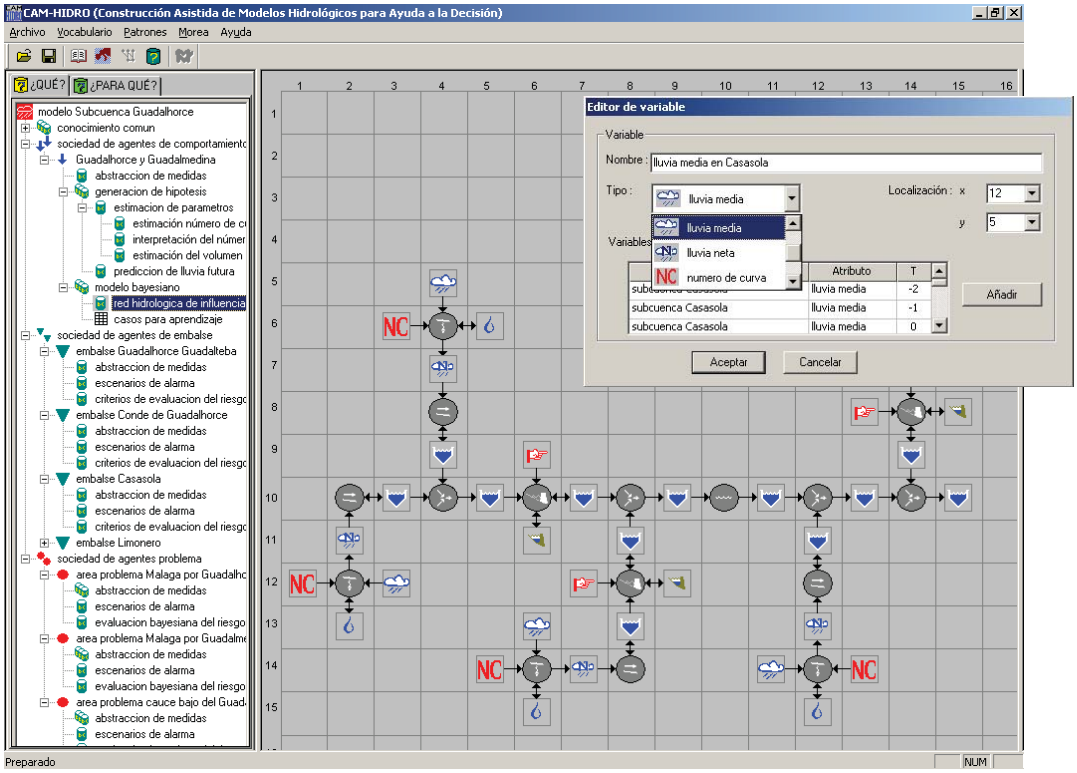


Figura 7.12: Ejemplo de pantalla mostrada por la herramienta CAM-HIDRO para ayuda a la construcción de modelos SAIDA.

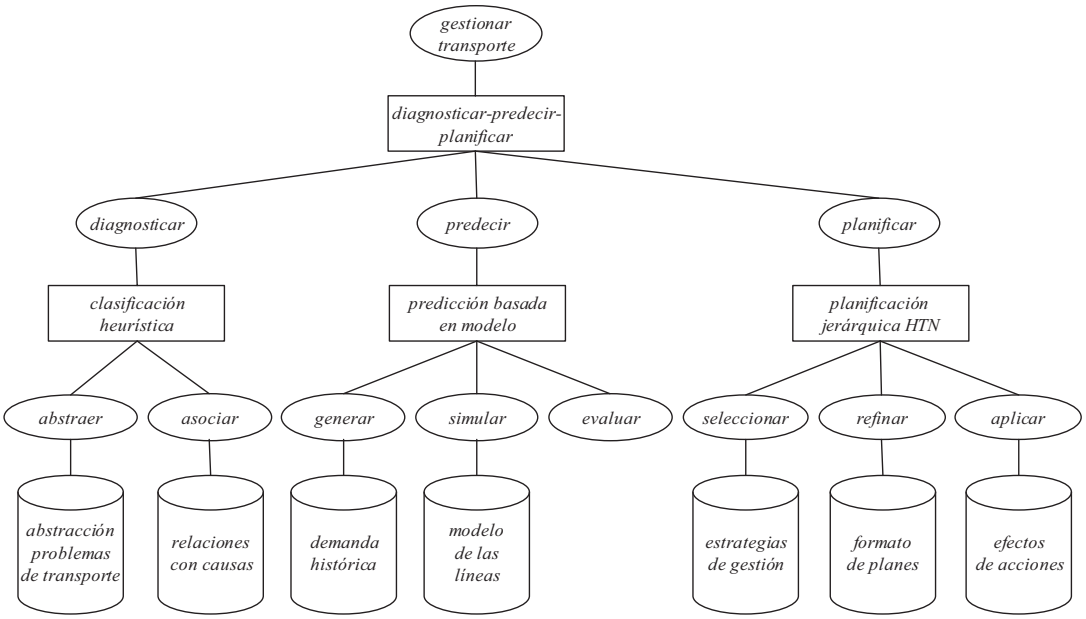


Figura 7.13: Estructura de tareas, métodos y bases de conocimiento del sistema de ayuda a gestión de líneas de autobuses.

7.2.3 Gestión de transporte público

La figura 7.13 muestra la estructura de tareas, métodos y bases de conocimiento de un sistema para ayuda a la gestión de líneas de autobuses (realizado inicialmente para la ciudad de Turín [Hernández et al., 04] y Vitoria [Molina, 05]). Este sistema hace uso de técnicas clasificativas para interpretación y diagnóstico además de un planificador heurístico jerárquico para propuestas de acciones de control.

Los sistemas SAE (Sistema de Ayuda a la Explotación) permiten a los operadores de un centro de control de gestión de líneas de autobuses conocer en cada momento la localización de los vehículos. Además, debido a que el sistema SAE dispone de información sobre la planificación de los horarios de cada una de las líneas, el sistema es capaz de determinar automáticamente vía comparación de la posición actual de los vehículos y la posición planificada, los retrasos y/o adelantos existentes y con ello advertir al operador de dicha situación e, incluso, enviar al

conductor de forma automática un mensaje indicando el desajuste medido. Esta función del SAE se denomina regulación automática y puede operar sin intervención de los operadores en tiempo de explotación, permitiendo que dichos desajustes se resuelvan por los propios conductores de vehículos.

Sin embargo, en la operación usual con las líneas, es frecuente que los operadores del centro de control encuentren situaciones problemáticas en las que es necesario intervenir. Estas situaciones se dan, por ejemplo, cuando se producen condiciones en el tráfico ajenas a los conductores de autobuses (tal como una retención importante de tráfico, un corte por una manifestación, un accidente de tráfico, etc.), cuando se produce una avería en un determinado autobús, cuando se produce una demanda inesperadamente alta de algún servicio (debido a algún acontecimiento de tipo social no esperado), etc. En dichas situaciones, los responsables del centro de control deben idear en poco tiempo las soluciones proponiendo acciones tales como: incorporar un vehículo de reserva en una línea, indicar a un conductor de un vehículo que realice un itinerario alternativo para superar un problema de falta de continuidad de una línea, enviar una grúa para reparar una avería de autobús, etc. Este tipo de acciones suelen ir en grupo, involucrando a varios vehículos, y deben estar adecuadamente coordinadas mediante posiciones y localizaciones. Para mejorar la respuesta de los operadores ante estos eventos, es útil contar con servicios por parte del sistema SAE que sugieran las acciones de control a realizar. A esta función se le denomina manejo de incidencias en donde, a diferencia del caso anterior, las soluciones no se establecen mediante el reajuste de la velocidad de los vehículos en las líneas, sino que deben realizarse acciones de mayor envergadura que normalmente involucran a diferentes vehículos y/o la ruptura del comportamiento normal.

El papel del sistema SAE ante este tipo de situaciones no es el de un sustituto del operador para llevar a cabo de forma automática la recuperación del problema, debido a la responsabilidad que supone la decisión de la solución, además de que su

puesta en práctica normalmente exige la participación de diversas personas. El papel del sistema SAE en estas situaciones debe entenderse como el de un asistente del operador que sugiere o recomienda planes de acciones, realizando los cálculos (tales como la hora a la que deben encontrarse dos vehículos, o el camino exacto que debe recorrer un vehículo de reserva para incorporarse en un determinado punto en una línea con problemas de servicio) y justifica adecuadamente su propuesta para que el operador pueda asumir la responsabilidad de su realización.

Para funcionar de la forma descrita, el sistema opera en tiempo real integrado en el sistema general de adquisición y monitorización. El sistema de recomendación de acciones recibe como información de entrada la configuración de las líneas, la localización de los vehículos e información acerca de tiempos de viaje. Como resultado, genera los problemas detectados y recomendación de acciones. El sistema de información monitoriza la localización de cada uno de los vehículos de las líneas, lo que permite determinar de forma automática los retrasos o desajustes existentes lo que entiende como síntomas de los problemas posibles. A partir de esta información el sistema es capaz de determinar la gravedad de la situación agregando información relativa a varios vehículos para identificar grupos retrasados, retraso de un vehículo individual, etc. y estima el impacto que tiene dicho problema en el servicio.

Los problemas considerados incluyen, diversos tipos de retrasos: retrasos individuales referidos a un solo autobús (leve, medio, importante) y retrasos generalizados referidos a varios autobuses. Además, se considera también el problema de saturación de un recorrido debido a que los autobuses que lo realizan van al límite de su capacidad por exceso de demanda. Los problemas llevan asociada además una medida del impacto, lo que supone una estimación del efecto que en el servicio tiene cada problema. Por ejemplo, el sistema puede dar como resultado un problema en la línea Zaramaga “retraso generalizado en los autobuses A209 y A207 con impacto de 75 personas 12 minutos”.

Tras la identificación y clasificación de los problemas en las líneas, el sistema propone acciones dirigidas a resolverlos. Para ello, el sistema recomienda la realización de un plan expresado mediante un lenguaje que hace uso de un conjunto de primitivas de control y que se presenta adecuadamente a través de la interfaz de usuario del sistema. Cada una de estas primitivas se entiende como una acción atómica que debe transmitir el operador del centro de control a un determinado conductor de un vehículo (autobús en la línea, vehículo de refuerzo, etc.). Un plan para resolver un problema se expresa mediante una secuencia de acciones de este tipo. El método de recomendación automática de acciones de control, ante la presencia de problemas en una línea de autobuses, propone uno o más planes solución como una secuencia ordenada de acciones atómicas

Tipos	*Formato de acciones*
orden a un autobús de línea	`ordenar regulación de tiempo al autobús <A>`
	`ordenar envío en vacío al autobús <A> a la parada <P>`
	`ordenar recorrido express al autobús <A>`
	`ordenar limitación de recorrido al autobús <A>`
orden a un autobús de refuerzo	`ordenar realizar refuerzo al autobús <A> desde la parada <P>`
	`ordenar realizar refuerzo al autobús <A> desde cabecera de línea`
notificación de acción	`notificar refuerzo al autobús <A> con especial <E> desde la parada <P>`
cambio de horario	`cambiar horario por rotación en la línea <L>`
	`cambiar de regulación por horario a frecuencia en la línea <L>`
	`cambiar de regulación por frecuencia a horario en la línea <L>`
	`cambiar el horario de referencia en la línea <L>`

Figura 7.14: Ejemplos de acciones atómicas contempladas por el sistema.

La figura 7.14 muestra el conjunto de acciones atómicas que considera el sistema. Dentro de dicho conjunto, hay acciones que comienzan por la palabra ordenar que indican ordenar al conductor de un vehículo una determinada acción. Estas acciones se dividen, por un lado, en las que están dirigidas a un autobús en línea (por ejemplo, realizar un recorrido express o una limitación para autobuses) y, por otro, acciones dirigidas a realizar un determinado refuerzo a autobuses especiales. Las

acciones que comienzan por la palabra notificar (por el momento sólo una) están orientadas a notificar una información a un conductor de un autobús. Finalmente, las acciones que comienzan por la palabra cambiar están dirigidas al propio operador para realizar un cambio en el horario de la propia línea con ayuda del sistema de información (por ejemplo, realizar una rotación de horario).

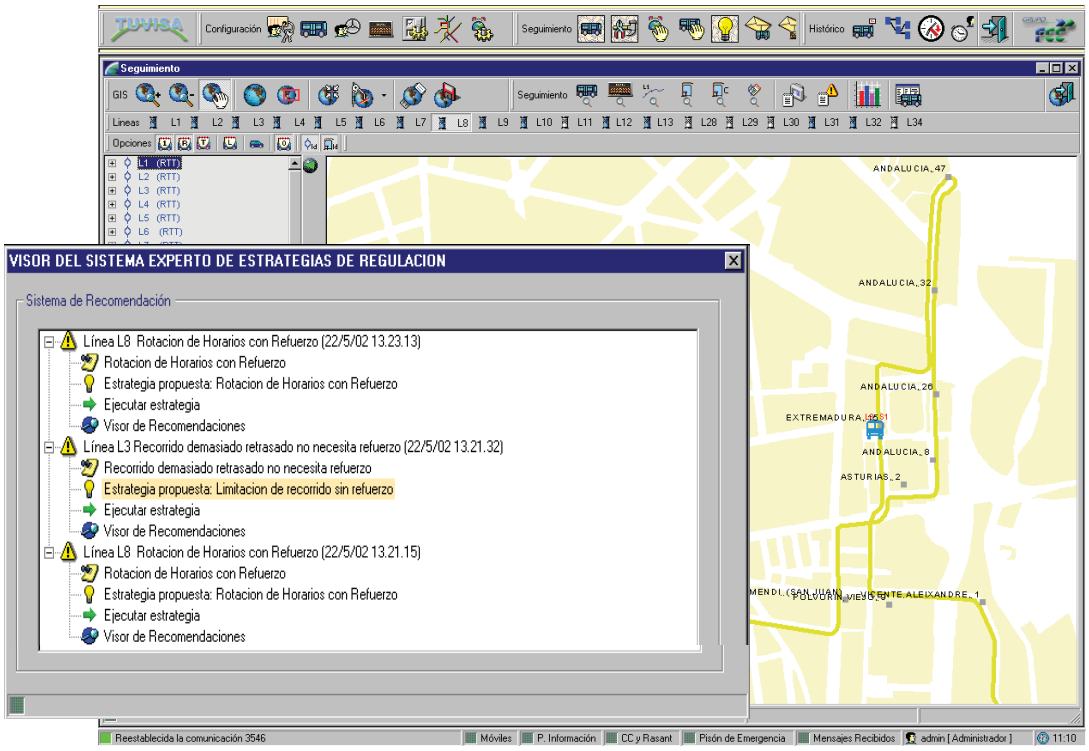


Figura 7.15: Ejemplo de plan recomendado por el sistema.

El sistema propone planes para cada línea dirigidos a resolver los problemas detectados. Por ejemplo, para la línea Zaramaga puede ser:

1. ordenar envío en vacío al autobús 20 a la parada Reyes de Navarra, 50
2. ordenar realizar refuerzo al autobús Especial N°7 desde la parada Cofradía de Arriaga, 5
3. notificar refuerzo al autobús 20 con especial R7 desde la parada Cofradía de Arriaga 5

Para la construcción del sistema de recomendación automática de acciones para líneas de autobuses se utilizan dos tipos de métodos de resolución de problemas principalmente: *clasificación heurística*, para diagnóstico del estado de las líneas, y *planificación jerárquica HTN*, para propuesta de recomendaciones. En los próximos apartados se describe cada uno de estos métodos.

Para llevar a cabo el proceso de diagnóstico de problemas de las líneas, como paso previo a la fase de recomendación, se utiliza clasificación heurística. Este método es apropiado en el área de transporte público para detectar y clasificar los problemas existentes en las líneas de autobuses. En concreto, en la fase de abstracción, los datos procedentes de la localización de las líneas, comparados con la localización esperada en cada momento proporciona los desajustes de servicio que se consideran las observaciones o síntomas de los problemas. El proceso de abstracción consiste en interpretar adecuadamente dichos desajustes para agregar la información para varios vehículos, clasificar el tipo de retraso, etc. Para ello, es se utilizan una serie de reglas, caracterizados convenientemente sobre las clases de escenarios problema que hay en las líneas. Los datos utilizados no son sólo las posiciones de los autobuses en cada momento sino que se manejan también ciertos códigos de mensajes enviados por los conductores (por ejemplo, alarma de llenado del autobús). Seguidamente, en la fase de asociación se hacen hipótesis de las posibles causas, solicitando si es necesario información adicional a los operadores, para determinar la clase de problema (avería de vehículo, dificultades de circulación en una zona, etc.). Por ejemplo, el sistema maneja la siguiente regla que utiliza una sintaxis propia para luego traducirse a CLIPS prácticamente de forma directa:

```
SI <A1> es autobus, nombre de <A1> es <NA1>,
   gravedad_del_retraso de <A1> es media,
   linea de <A1> es <L>,
   tipo_servicio de <A1> es regulación,
   estado_horario de <A1> es retrasado,
   circulacion de <L> es no_reforzada
   retraso_relativo de <A1> es unico
ENTONCES crear <P> como problema,
         clase de <P> es retraso_individual_medio,
         autobus_causante de <P> es <NA1>, linea de <P> es <L>
```

Para evaluar la gravedad del problema el sistema realiza una estimación del impacto que tienen los problemas en curso realizando una predicción de la falta de servicio que se va a realizar. Así, puede ser más grave un retraso de 5 minutos en una línea donde en cada parada suele haber una media de 12 personas esperando, que en otra en la que el retraso es de 10 minutos pero el número medio de personas en cada parada es de 2. Esto supone realizar un proceso de predicción que realiza los siguientes tres pasos:

1. *Estimar* la demanda futura de las paradas. Para realizar dicha estimación puede ser suficiente con una aproximación que hace uso de conocimiento sobre demanda histórica de servicio. Dicha demanda histórica contiene el número medio de personas que esperan en cada parada para cada tramo horario (con la precisión necesaria en cada momento). Este valor puede provenir de alguna base de datos histórica, mediante una muestra estadística realizada o, si no se dispone de información, puede ser directamente el experto el que realice la estimación.

2. *Simular* el recorrido de los autobuses por las líneas para los próximos T minutos (por ejemplo T = 15). Con ello se obtienen las paradas afectadas en los próximos minutos lo que requiere un proceso de simulación sencillo que utiliza como conocimiento la forma de las líneas (paradas, distancias) y la posición actual de los autobuses, teniendo en cuenta además los problemas existentes.

3. *Evaluar* el impacto del retraso. Después de los pasos anteriores, se conoce las paradas afectadas y una estimación de p_i, el número de personas esperando en la parada *i*. Además se sabe e_i, es decir es el incremento del tiempo de espera en dicha parada debido al retraso del autobús. El cálculo del impacto se realiza mediante el siguiente proceso. La medida de impacto *I* se calcula como:

$$I = \sum_{i=1}^{n} p_i \cdot e_i$$

La medida de impacto se puede expresar directamente como *I*, lo que facilita la comparación de problemas en diferentes líneas o, para dar una visión más intuitiva, se puede desdoblar en forma de <*P* personas, *E* minutos> en donde *E* es el tiempo medio de espera en esas paradas y se obtiene como *E = I/P* siendo *P* el número total de personas que esperan en las paradas:

$$\mathrm{P} = \sum_{i=1}^{n} p_i$$

La función de recomendación de acciones de control para resolver los problemas detectados se apoya en el método de *planificación jerárquica HTN.* Este método se basa en que la construcción del plan es un problema de diseño que se resuelve mediante la intervención de varias etapas de diseño a diferentes niveles de abstracción dentro de una jerarquía de decisiones. El método utiliza una representación organizada en una jerarquía de especialistas similar a la que se plantea en el sistema AIR-CYL. Este método es muy útil en donde la complejidad del problema exige una adecuada modularización de los criterios de planificación en módulos especializados en ciertos conocimientos heurísticos como es el caso de recomendación de acciones en gestión de líneas de autobuses. En este caso se tiene una jerarquía de especialistas en donde el de más alto nivel es el encargado de las funciones de recomendación globales de una determinada línea de autobuses. Dicho especialista es capaz de proponer acciones generales de solución a problemas que deben ser confirmadas y/o detalladas por los especialistas de más bajo nivel, por ejemplo especialistas de envío en vacío, especialista en refuerzo, regulación, etc.

El conocimiento utilizado, de acuerdo con el tipo de método, es (1) especialistas responsables de metas, (2) conveniencia de planes, (3) formato de planes. La figura 7.16 muestra ejemplos del conocimiento de conveniencia de planes para el

especialista de una línea en donde se relacionan clases de problemas existentes en la línea con clases de acciones que pueden resolver dichos problemas. El conocimiento se expresa en reglas en donde, en el antecedente, se escriben condiciones sobre los problemas y, en el consecuente, las clases posibles de planes de actuación.

```
SI <X> es problema, clase de <X> es retraso_individual_leve,
   autobus_causante de <X> es <A>
ENTONCES  crear <P> como plan, clase de <P> es regulacion,
          Autobus afectado de <P> es <A>
SI <X> es problema, clase de <X> es retraso_individual_leve,
   autobus_causante de <X> es <A>
ENTONCES crear <P> como plan, clase de <P> es recorrido_express,
         autobus afectado de <P> es <A>
SI <X> es problema, clase de <X> es retraso_individual_severo,
   autobus_causante de <X> es <A>
ENTONCES crear <P> como plan, clase de <P> es regulacion,
         autobus afectado de <P> es <A>
SI <X> es problema, clase de <X> es retraso_individual_severo,
   autobus_causante de <X> es <A>
ENTONCES crear <P> como plan, clase de <P> es envio_vacio_con_refuerzo,
         autobus afectado de <P> es <A>
SI <X> es problema, clase de <X> es retraso_individual_severo,
   autobus_causante de <X> es <A>
ENTONCES crear <P> como plan, clase de <P> es envio_vacio,
         autobus afectado de <P> es <A>
SI <X> es problema, clase de <X> es retraso_generalizado,
   autobus_causante de <X> es <A>, linea de <X> es <L>,
   <Y> es autobus, nombre de <Y> es <A>, linea de <Y> es <L>,
   tipo_regulacion de <L> es frecuencia,
   retraso de <Y> es <T>, <T> <= 5,
ENTONCES crear <P> como plan, clase de <P> es nueva_por_frecuencia,
         linea de <P> es <L>
SI <X> es problema, clase de <X> es retraso_generalizado,
   autobus_causante de <X> es <A>, linea de <X> es <L>,
   <Y> es autobus, nombre de <Y> es <A>, linea de <Y> es <L>,
   retraso de <Y> es <T>, <T> > 5,
ENTONCES crear <P> como plan,
         clase de <P> es rotacion_horarios_con_refuerzo,
         autobus afectado de <P> es <A>, linea de <P> es <L>
SI <X> es problema, clase de <X> es retraso_generalizado,
   autobus_causante de <X> es <A>, linea de <X> es <L>,
   <Y> es autobus, nombre de <Y> es <A>, linea de <Y> es <L>,
   retraso de <Y> es <T>, <T> > 5,
ENTONCES crear <P> como plan, clase de <P> es rotacion_horarios,
         autobus afectado de <P> es <A>, linea de <P> es <L>
SI <X> es problema, clase de <X> es retraso_generalizado,
   autobus_causante de <X> es <A>, linea de <X> es <L>,
   <Y> es autobus, nombre de <Y> es <A>, linea de <Y> es <L>,
   tipo_regulacion de <L> es horario,
   retraso de <Y> es <T>, <T> <= 5,
ENTONCES crear <P> como plan, clase de <P> es regulacion_por_frecuencia,
         linea de <P> es <L>
SI <X> es problema, clase de <X> es saturacion_de_recorrido,
   autobus_causante de <X> es <A>
ENTONCES crear <P> como plan, clase de <P> es refuerzo,
         autobus afectado de <P> es <A>
```

Figura 7.16: Ejemplo parcial de base de conocimiento para selección de planes

Existe una regla para cada clase de plan que se puede llevar a cabo, aunque podría darse el caso de que para diferentes problemas se llevase a cabo el mismo plan. También es normal que ante una clase de problema, existan varios planes alternativos que puedan llevarse a cabo para resolver el problema, de forma que si uno de ellos no se puede realizar se intente con otro. Esto se decidirá en alguna fase posterior. Tendrá mayor prioridad el primer plan que aparezca en una regla. Por ejemplo, si un problema es de retraso individual leve, la primera regla indica que se debe tomar un plan de regulación mientras que la segunda, ante esa misma condición, recomienda un recorrido express. Esto quiere decir que hay dos planes posibles y que si no se puede solucionar el problema mediante regulación, entonces se debe hacer un recorrido express. También cabe la posibilidad de modificar las prioridades de los planes añadiendo un atributo (por ejemplo prioridad) al concepto plan, que permita de forma explícita indicar cuál es el primer plan que hay que intentar llevar a cabo.

En el caso de un retraso individual medio, se dan tres posibles planes: regulación, limitación con refuerzo y limitación. Si el retraso individual es severo, los posibles planes son: regulación, envío en vacío con refuerzo y envío en vacío. En conclusión, en el caso de retrasos individuales siempre se propone como plan realizar una regulación, que es la forma más sencilla de solucionar el problema, y además, dependiendo de la gravedad del retraso se propone recorrido express, limitación o envío en vacío, de menor a mayor retraso. En los dos últimos casos se introduce un refuerzo si es posible.

En el caso de un retraso generalizado se propone uno u otro plan en función del retraso. Si el retraso es pequeño (en el ejemplo <= 5 minutos) entonces si se está regulando por horario se pasa a regular por frecuencia y si se está regulando por frecuencia se propone cambiar dicha frecuencia. En cambio con un retraso grande (> 5 minutos) se proponen dos posibles planes: realizar una rotación de horarios incluyendo un refuerzo o, si no es posible, realizar sólo la rotación.

El conocimiento sobre refino de planes tiene para cada clase de plan las acciones que hay que realizar para llevarlo a cabo. La forma de obtener las acciones que forman el plan no es directamente en un solo paso a partir de la clase de plan sino que el plan se va refinando por medio de los especialistas. Así, en un primer paso el plan se descompone en acciones atómicas y en acciones abstractas, que deben ser refinadas por otros especialistas en pasos posteriores. El conocimiento representado en esta base es el conjunto de acciones (concretas o de refino) en que se descompone cada plan. Por ejemplo, si el plan es recorrido express, entonces las acciones que lo componen es:

```
LINEA EN CURSO <L>
AUTOBUS AFECTADO <A>
PARADA AFECTADA <P>
PLAN recorrido_express
FORMATO
  ACCION CONCRETA ordenar recorrido express al autobus <A>
  ACCION ABSTRACTA reforzar_desde_parada
  ACCION CONCRETA ordenar realizar refuerzo al autobus <A> desde parada <P>
```

La primera acción es concreta, mientras que la segunda ha de ser refinada por un especialista. Este especialista construirá su plan, haciendo uso también de otros especialistas, y sustituirá esa acción abstracta por el conjunto de acciones que componen el subplán. Los especialistas de bajo nivel utilizan soluciones algorítmicas para tomar decisiones sobre la forma de refinar los planes. Por ejemplo, el especialista de envío en vacío aplica un algoritmo de búsqueda sobre la línea que encuentra la parada a la que hay que enviar en vacío el autobús retrasado teniendo en cuenta que dicha parada debe ser la primera a la que pueda llegar en horario.

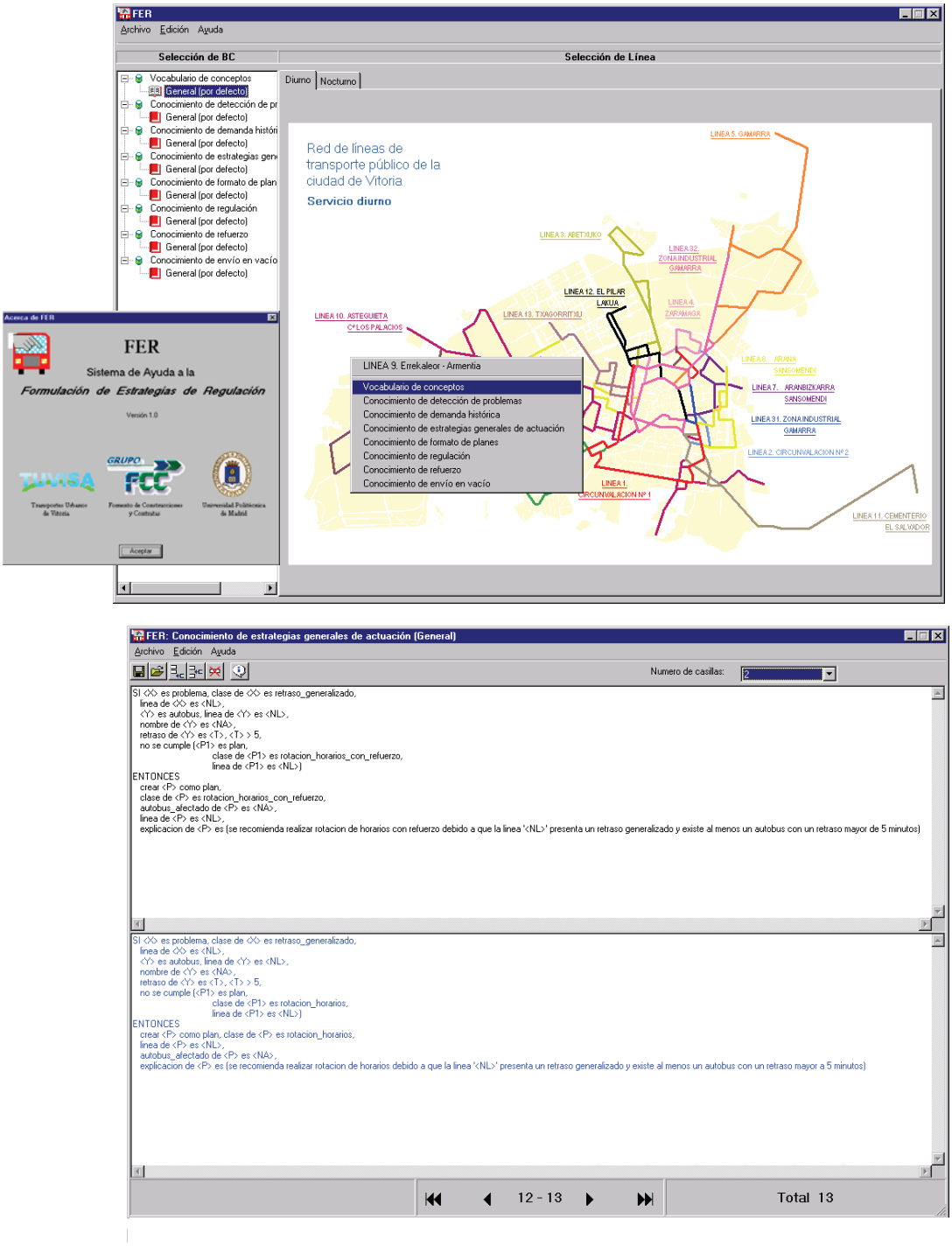


Figura 7.17: Ejemplos de ventanas mostradas por el sistema FER para ayuda a la construcción de modelos de gestión de líneas de autobuses.

Con el fin de facilitar la construcción y mantenimiento de los modelos de gestión de líneas de autobuses, se desarrolló una herramienta denominada FER (sistema de ayuda a la Formulación de Estrategias de Regulación) que permite a un operador no informático acceder a los contenidos de las bases de conocimiento para su consulta y modificación. El sistema ofrece una forma intuitiva de acceso a las diferentes bases de conocimiento con ayuda de mapas de ubicación de las líneas, además de editores especializados de los componentes de las bases de conocimientos que realizan comprobaciones de tipo sintáctico y semántico de las modificaciones realizadas. La figura 7.17 muestra ejemplos de ventanas del interfaz de usuario de dicha aplicación.

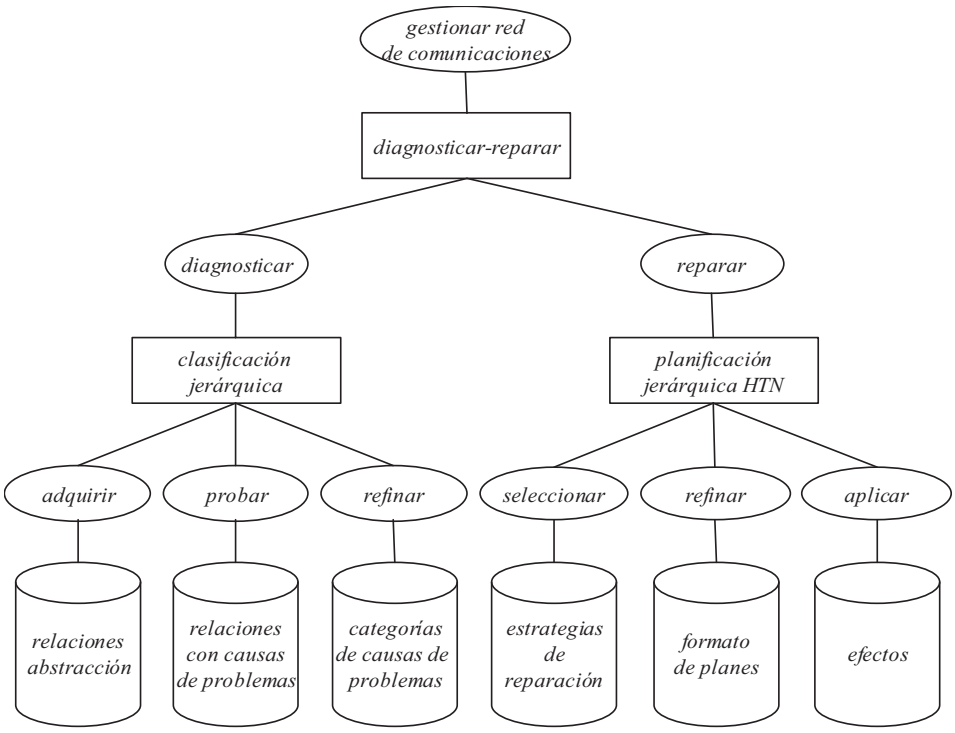


Figura 7.18: Estructura de tareas, métodos y bases de conocimiento de EXPERNET.

El sistema aquí presentado tiene como antecedente los trabajos realizados para el proyecto FLUIDS, un proyecto realizado por un consorcio europeo de grupos de investigación y empresas, orientado a la definición de métodos avanzados de interacción hombre máquina [Hernández et al., 04]. En dicho proyecto se realizó un prototipo demostrativo para la gestión de autobuses de la ciudad de Turín (Italia). Parte del modelo de gestión aplicado a Turín fue utilizado como base para el desarrollo de una solución para la ciudad de Vitoria para empresa de transportes TUVISA (Transportes Urbanos de Vitoria), en la que se hizo un refino y adaptación del modelo general de Turín además de construir una solución eficiente de implementación con el fin de operar en tiempo real embebido en el software del centro de control de gestión de autobuses [Molina, 05]. Dicho trabajo fue realizado por el Departamento de Inteligencia Artificial de la Universidad Politécnica de Madrid para la empresa SCC del grupo FCC (Fomento de Construcciones y Contratas) quien fue la responsable del suministro del software global del centro de control a TUVISA.

7.2.3 Otros sistemas

La figura 7.18 muestra una parte de otro ejemplo para el caso de g*estión de red de comunicaciones.* Se trata de un sistema desarrollado en el proyecto EXPERNET con financiación europea en un consorcio internacional para ayuda la administración de redes de comunicaciones en la red nacional de Ucrania [Molina, Ossowski, 99] (figura 7.19). En este problema el objetivo es asistir a los administradores de nodos de la red de comunicaciones en tareas tales como diagnosticar problemas realizando acciones exploratorias y sugerir planes de acciones de reparación de los fallos detectados. Por ejemplo, los síntomas observados pueden ser: un cierto servicio como ftp, www, etc. no responde, no se puede acceder a un host, sobreutilización o infrautilización de enlaces o equipos, etc. Las causas pueden ser capacidad inadecuada de un recurso, desequilibrio de

carga de trabajo, averías de equipos, etc. Las acciones de reparación pueden consistir en reconfiguración y/o sustitución de enlaces y equipos. En la construcción de este sistema se combinaron dos métodos (clasificación jerárquica y planificación jerárquica HTN) como muestra la figura. Además se utilizó una solución multiagente para distribuir el conocimiento en los diversos nodos de la red.

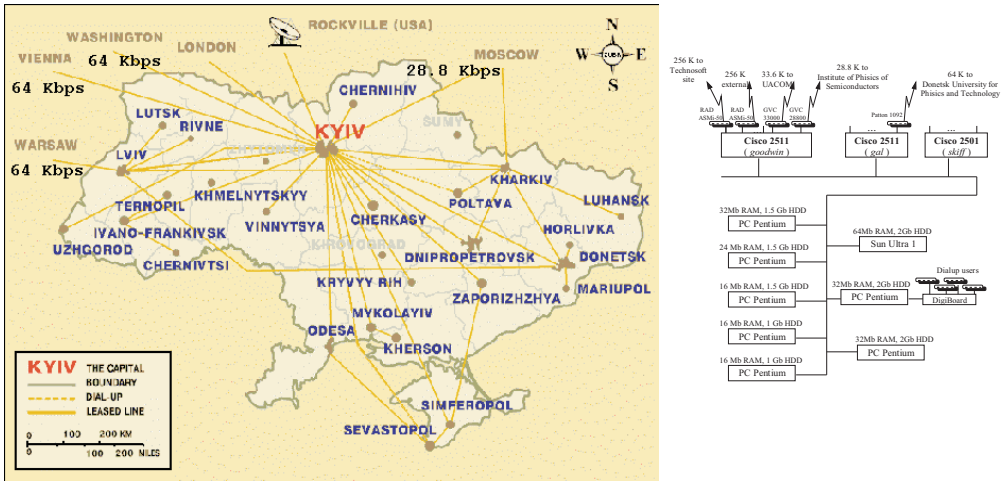


Figura 7.19: Dominio de operación del sistema de EXPERNET.

Especialmente significativo fue el hecho de identificar en una fase temprana del proyecto que los expertos ucranianos realizaban el diagnóstico de una forma análoga al método de clasificación jerárquica. Ello facilitó de forma importante el trabajo de adquisición del conocimiento teniendo en cuenta que, en el contexto internacional de desarrollo del proyecto, esta actividad hubo que hacerla con ciertas limitaciones con un número reducido de entrevistas y comunicación vía *email*. Respecto a la parte de planificación, se utilizaron especialistas relacionados con la gestión del rendimiento, gestión de la configuración de la red y la gestión de las averías.

El proyecto EXPERNET se desarrolló fundamentalmente en un contexto de investigación y tenía como fin la construcción de un prototipo demostrativo de la viabilidad de soluciones con técnicas basadas en el conocimiento y organización

multiagente para el problema de administración distribuida de una red de comunicaciones. El prototipo desarrollado fue validado mediante instalación provisional en tiempo real con la red de Ucrania sobre la que se desarrolló el proyecto.

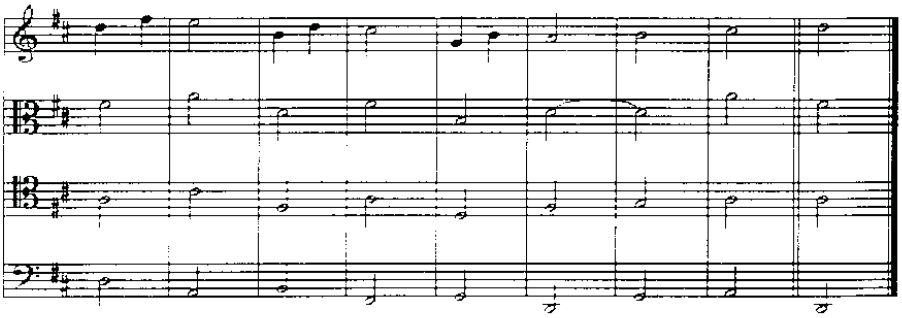


Figura 7.20: Ejemplo de armonización a cuatro voces.

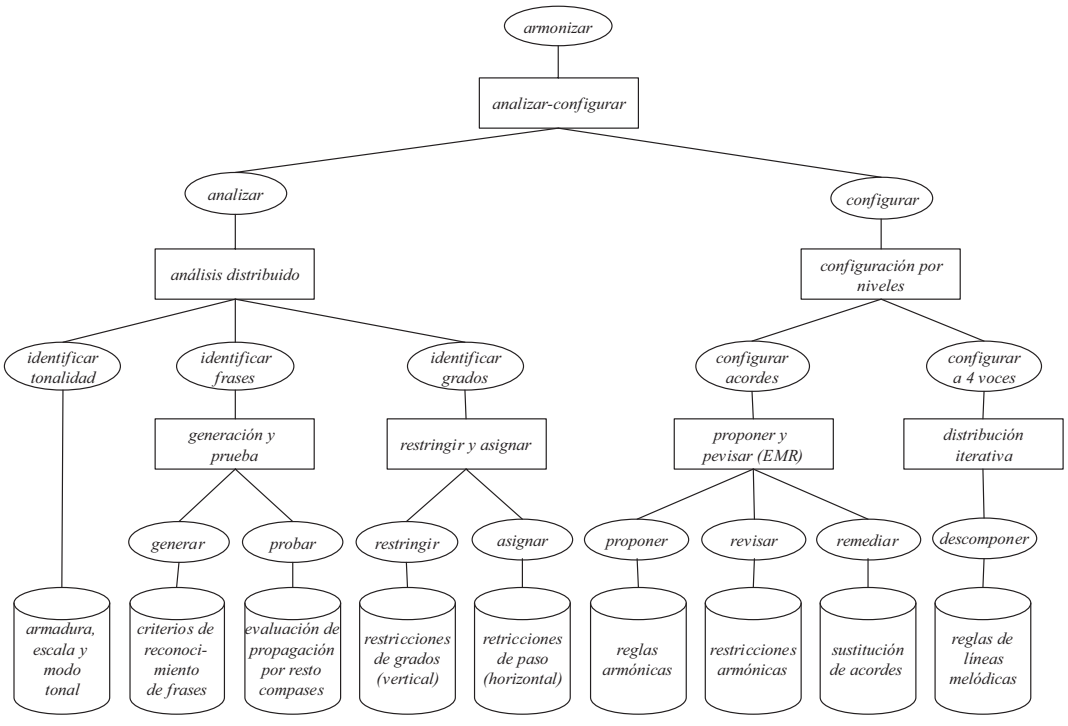


Figura 7.21: Estructura de tareas, métodos y bases de conocimiento del problema de armonización musical a cuatro voces.

Otro ejemplo de combinación de métodos se da en el campo de composición musical. Dentro de este campo se encuadra el problema de armonización a cuatro voces que consiste en que, dada una melodía de referencia se crean tres melodías que superpuestas a la primera creen un determinado efecto armónico (figura 7.20). La figura 7.21 muestra la descomposición en tareas, métodos y bases de conocimiento correspondiente a este problema. En este caso, la tarea general, denominada armonizar, se descompone en dos tareas: una que tiene como fin el análisis de la melodía de referencia para extraer sus características principales desde el punto de vista de armonización y otra para la configuración de las otras tres melodías. La primera tarea supone la identificación de características en cuanto a tonalidad, frases sobre patrones rítmicos y melódicos, además de los grados correspondientes a las notas de la melodía. La segunda tarea lleva a cabo el proceso de configuración haciendo uso del método de proponer y revisar para proponer acordes con el añadido de la distribución en líneas melódicas. Sobre este problema pueden consultarse los trabajos de Ignacio Parres y Jaime Thos [Parres, 98; Thos, 01].

En resumen, estos ejemplos y los mostrados en las secciones anteriores muestran la utilidad de la forma de llevar a cabo organización y uso del conocimiento que plantean los métodos de resolución de problemas. En dichos ejemplos, dada la complejidad de los modelos, resulta eficaz hacer uso de estas técnicas que permiten la descripción a mayores niveles abstracción con vistas complementarias sobre el modelo de conocimiento. Esta visión aporta como ventajas: (1) facilidades para estructurar y encapsular los componentes de un modelo, (2) entidades descriptivas meta-conocimiento para incorporar niveles reflexivos (3) ayudas a la reutilización y compartición de conocimiento basadas en la posibilidad de uso de estructuras abstractas de conocimiento, (4) arquitecturas abiertas, que permiten la inspección del conocimiento del sistema, (5) un enfoque global para llevar a cabo el diseño de sistemas que integran tanto técnicas de Inteligencia Artificial con técnicas convencionales.

7.3. Ejercicios

EJERCICIO 7.1. En una temporada, una agencia de viajes oferta un conjunto total de 50 viajes turísticos prefijados. Cada viaje turístico incluye transporte, alojamiento, recorridos, etc. El total de la oferta incluye destinos de diversas clases incluyendo, por ejemplo, viajes de tipo cultural, aventura, de ocio, diseñados para jóvenes, familias o tercera edad, etc. El problema consiste en encontrar el viaje turístico (o viajes) que mejor encajan con las características de un determinado cliente, teniendo en cuenta aspectos tales como su edad, coste máximo, preferencias, tiempo disponible, etc.

SE PIDE:

Indicar de forma razonada la posibilidad de utilizar en el problema anterior cada uno de los métodos de resolución de problemas estudiados.

EJERCICIO 7.2: Una empresa que comercializa equipos informáticos debe determinar las características de un determinado equipo en función de las necesidades de cada cliente, eligiendo de acuerdo con dichas necesidades la CPU del ordenador, el tamaño de la memoria, el espacio en disco, el tipo de impresora, velocidad de impresión, etc.

SE PIDE:

Indicar de forma razonada la posibilidad de utilizar en el problema anterior cada uno de los métodos de resolución de problemas estudiados.

EJERCICIO 7.3: Considerar los siguientes problemas:

Problema (a): Un vendedor de automóviles debe encontrar el vehículo más adecuado para un determinado cliente. Para ello el vendedor, a partir de cierta información inicial, interroga al cliente sobre sus preferencias personales, disponibilidad económica, etc. con el fin de seleccionar la clase de vehículo más apropiado.

Problema (b): Con objeto de facilitar los futuros viajes espaciales (tales como el planeado viaje a Marte) se desea disponer de herramientas automáticas que ayuden a los astronautas a identificar minerales. Para ello, en este contexto, se plantea el problema de identificar la clase de roca a partir de las características observadas tales como dureza, color, rugosidad, resistencia a fractura, solubilidad, etc.

Problema (c): Se trata de determinar los cultivos más adecuados para un conjunto de parcelas teniendo en cuenta características climáticas, la calidad del suelo, últimos cultivos realizados en cada parcela, restricciones de cultivos entre parcelas contiguas, etc. Como resultado se debe producir el conjunto de cultivos en las parcelas.

Problema (d): La terapia para un paciente de una determinada enfermedad consiste en aplicar en pasos sucesivos, siguiendo un determinado orden, la administración de determinadas sustancias químicas para diferentes facetas de la enfermedad (que cubren tanto síntomas como las propias causas). El problema consiste en determinar (1) qué sustancias químicas deben administrarse, (2) cantidad de cada una de las sustancias, (3) en qué orden se deben administrar.

Problema (e): Se trata de diagnosticar enfermedades del ganado vacuno. A partir de síntomas observados tales como problemas de alimentación, dificultades de mantenimiento de equilibrio, fiebre, etc. se trata de determinar la posible enfermedad realizando, en su caso, pruebas analíticas adicionales.

Problema (f): Para aconsejar la rama de estudios más adecuada para un estudiante se cuenta con un conjunto datos recogidos en un formulario, en donde se dispone de información relativa a los resultados de diversos test psicológicos, preferencias personales, etc. El problema consiste en encontrar la rama de estudios más adecuada del estudiante utilizando exclusivamente la información recogida en dicho formulario.

Problema (g): A partir de la presencia de problemas de retrasos en una red de ferrocarriles, se trata de encontrar la secuencia de acciones que pueden eliminar dichos retrasos. Dichas acciones incluyen medidas de control tales como: cortar el tráfico en determinado punto, enviar máquinas de reserva, desviar a un tren por una vía alternativa, etc.

SE PIDE:

Indicar de forma razonada la posibilidad de resolver los problemas anteriores mediante alguno (o algunos) de los métodos estudiados. Dentro de cada problema, para el método o métodos seleccionados indicar (1) el conocimiento del problema requerido para poder aplicar el método y (2) la forma de interacción con el usuario.

EJERCICIO 7.4. Considerar los siguientes problemas:

(a) *Seguridad en astilleros.* El departamento de una empresa de construcción naval especializado en los aspectos de seguridad de los trabajadores debe determinar si la obra que se está realizando en un astillero para construir un determinado barco de transporte de mercancías satisface los requisitos de seguridad establecidos por la normativa actual.

(b) *Gestión de patrimonio.* Un equipo de asesoría sobre inversiones es responsable de la supervisión y gestión del patrimonio económico de un determinado grupo familiar que se encuentra distribuido en fondos de inversión en renta fija, fondos en renta variable, inversiones en inmuebles, etc. Periódicamente dicho equipo (1) estudia la evolución reciente de las inversiones para determinar si su rentabilidad es aceptable y, en caso contrario, (2) de acuerdo con las predicciones del mercado, toma decisiones sobre cambios en las inversiones con el fin de mejorar las expectativas de rentabilidad.

(c) *Incendios forestales*. Una empresa consultora en el área medioambiental está especializada en la realización de tareas específicas relacionadas con bosques y parques naturales. Una de las tareas que realiza dicha empresa es comprobar en qué grado una determinada zona forestal cumple los requisitos de defensa contra posibles incendios. Dichos requisitos pueden ser, por ejemplo, los que establece la normativa de una determinada región o comunidad autónoma. Para ello, se tienen en cuenta las características de la zona forestal tales como especies de árboles, la densidad de árboles en el bosque, proporción de matorral, zona geográfica, clima, presencia de cortafuegos, etc. así como datos sobre el equipamiento de vigilancia y defensa contra incendios tales como puestos de observación, distancia a equipos de bomberos, disponibilidad de maquinaria especializada, etc.

(d) *Fabricación de botellas de vidrio*. Para supervisar y asegurar el correcto funcionamiento de un proceso de fabricación de botellas de vidrio se realiza un control que permite detectar la presencia de fallos de construcción de las botellas en cuanto a dimensiones, peso, color, grosor, resistencia a la rotura, etiquetado, etc. El análisis de dichas anomalías y la búsqueda de información adicional permite encontrar los defectos en el proceso de fabricación y, con ello, sugerir cuáles son las soluciones cuando se presentan problemas.

(e) *Red de distribución de agua*. En una red de distribución de agua se manejan diversos componentes tales como depósitos, canalizaciones, motores de bombeo, etc. cuyo estado es conocido en un centro de control mediante sensores que miden el caudal y/o nivel en los diversos puntos. Con ayuda de actuadores remotos se abren y cierran canalizaciones y se ponen en funcionamiento los motores de bombeo que llenan y vacían los depósitos. El objetivo de los responsables de la gestión de la red es (1) actuar periódicamente sobre los órganos de control (motores de bombeo, desvíos de canalizaciones, etc.) para garantizar el abastecimiento de agua con un coste de energía eléctrica mínimo, (2) comprobar que la red opera correctamente asegurándose de que no se producen fugas de agua por fisuras o roturas de canalizaciones, problemas de funcionamiento de motores, etc. (esto puede requerir la realización de medidas y exploraciones de la red in situ ante la sospecha de una determinada fuga u otro problema) (3) plantear formas de reparación de las averías seleccionando los puntos de corte en la red, actividades a realizar, personal para llevar a cabo la reparación, etc.

SE PIDE:

Indicar de forma razonada la posibilidad de resolver los problemas anteriores mediante los diferentes métodos de resolución de problemas (proponer y revisar, clasificación jerárquica, etc.). Considerar la posibilidad de combinar métodos (total

o parcialmente). Dentro de cada caso indicar (1) tipos de tarea o tareas que se realizan indicando de forma clara sus roles de entrada y de salida, (2) el método o métodos utilizados justificando su uso, (3) el contenido de las bases de conocimiento mostrando ejemplos sencillos y (4) ejemplos parciales de razonamiento que ilustren la inferencia y la forma de interacción con el usuario del sistema.

8 Metodologías y herramientas

En el presente capítulo se describe la utilización de los métodos bajo la perspectiva global de la ingeniería del conocimiento haciendo referencia a metodologías de construcción de sistemas inteligentes y herramientas informáticas de ayuda al desarrollo.

8.1 Metodologías de ingeniería del conocimiento

La *ingeniería del conocimiento* es una disciplina dentro de la informática que reúne un conjunto de métodos, técnicas y herramientas orientadas a sistematizar el proceso de construcción de un sistema de conocimiento. Ello facilita la evaluación de costes de trabajo a priori y el aprovechamiento entre distintos profesionales de experiencia en desarrollos previos para reducir esfuerzos y costes de desarrollo. La ingeniería del conocimiento maneja modelos de ciclo de vida de desarrollo de sistemas que tienen ciertas similitudes y diferencias con respecto a los manejados en informática convencional. Las etapas tradicionales en la construcción de un sistema basado en el conocimiento se pueden resumir en los siguientes pasos:

1. Estudio de viabilidad e impacto en la organización
2. Análisis del conocimiento
3. Diseño de la arquitectura software
4. Programación
5. Validación
6. Implantación
7. Mantenimiento

El desarrollo de un sistema inteligente en cierta forma incluye también la realización de estas actividades. No obstante su desarrollo presenta ciertas particularidades frente a los sistemas convencionales. Por ejemplo, en la etapa de estudio de viabilidad debe tenerse en cuenta que la realización de un sistema inteligente exige que el conocimiento de resolución de problemas normalmente basado en una o más personas expertas debe existir y ser accesible.

Respecto a la etapa de análisis del conocimiento, este proceso se observa no como el análisis de procedimientos de tratamiento de información sino como un proceso más complejo que se describe haciendo ciertas distinciones entre categorías de conocimiento. Esta etapa es crucial en la construcción de sistemas inteligentes y habitualmente presenta más complicaciones que las que se tienen en los sistemas convencionales dado que se tratan problemas semi-estructurados (no bien formalizados). En esta etapa, es necesario normalmente considerar fases de adquisición del conocimiento con ayuda de diferentes técnicas. Dichas técnicas pueden ser por ejemplo (1) de elicitación de conocimiento mediante entrevistas estructuradas, análisis de protocolos, etc. y (2) utilización de modelos abstractos para clases de problemas que guían el proceso de adquisición.

Como resultado de la etapa de análisis se produce lo que se denomina *modelo de conocimiento* que recoge los detalles sobre las formas de resolución de problemas y el conocimiento que interviene en ellas.

En la etapa de programación puede hacerse uso de herramientas especializadas y lenguajes de representación específicos propios del campo de la inteligencia artificial. Por ejemplo, lenguajes como Prolog (programación lógica), CLIPS (reglas de producción) o herramientas con una representación del conocimiento más avanzada.

La etapa de mantenimiento es clave en los sistemas basados en el conocimiento dado que tras ser implantados, las bases de conocimiento de dichos sistemas cuentan con una versión inicial de los criterios de resolución de problemas que, normalmente, sufrirán cambios a lo largo de la existencia del sistema. La arquitectura en forma de base de conocimiento explícita facilita el trabajo de mantenimiento que puede llevarse a cabo mediante personas encargadas de su puesta a punto (administradores de base de conocimiento) a medida que evoluciona el conocimiento de la organización.

El desarrollo de dichas etapas, en muchas ocasiones, no se realiza de forma lineal sino que requiere avances y retrocesos puntuales hasta alcanzar la versión final. Como alternativa a este enfoque existen otros modelos de ciclo de vida (basados en prototipos, de espiral, etc.) que suponen una variante dicho desarrollo lineal y que pueden ser más adecuados en situaciones que necesiten refinar los requisitos funcionales mediante ensayos de operación.

Una de las metodologías de ingeniería del conocimiento que incorpora los conceptos de análisis (tareas, métodos, inferencias, dominios, etc.) revisados en el presente texto es la metodología CommonKads. CommonKADS nació en la Unión Europea como resultado de varios proyectos consecutivos de investigación desde

1983 (KADS, KADS-II, KACTUS, TRACKS). CommonKADS se ha utilizado en una gran variedad de proyectos para desarrollo de sistemas comerciales. Empresas como Cap Gemini y Everest (sistemas en el sector financiero), empresas en Japón (incluído IBM), Touch Ross Management Consultants (fraudes en tarjetas), etc.

CommonKADS está dirigido al desarrollo de sistemas en general, aunque su punto fuerte se centra en el análisis de conocimiento. CommonKADS es aplicable también a gestión de conocimiento, análisis de procesos comerciales y análisis de requisitos.

CommonKADS considera la construcción de un sistema informático como la realización de tres clases de modelos:

1. *Modelo del contexto del sistema:*
    - Modelo de la organización: descripción del problema a resolver en la organización y evaluación de la viabilidad.
    - Modelo de las tareas: identificación de las tareas globales a realizar con entradas, salidas, precondiciones y criterios de rendimientos.
    - Modelo de agentes: identificación de quiénes realizan las tareas. Pueden ser humanos o sistemas. Se definen competencias, autoridad, etc.

2. *Modelo conceptual:*
    - Modelo del conocimiento: detalla el conocimiento utilizado para realizar las tareas (independiente de la implementación).
    - Modelo de comunicación: descripción de las transacciones entre los diversos agentes.

3. *Modelo de diseño:*

    Descripción computacional para llevar a cabo su implementación.

Entre dichos modelos destaca el modelo de conocimiento que permite abordar de una forma sistemática la construcción de sistemas basados en el conocimiento. CommonKADS aporta una solución relacionada con la idea de métodos de resolución de problemas, proporcionando medios descriptivos específicos que facilitan el análisis del conocimiento de un problema. La figura 8.1 resume las categorías del conocimiento consideradas a tres niveles: dominio, inferencia y tarea.

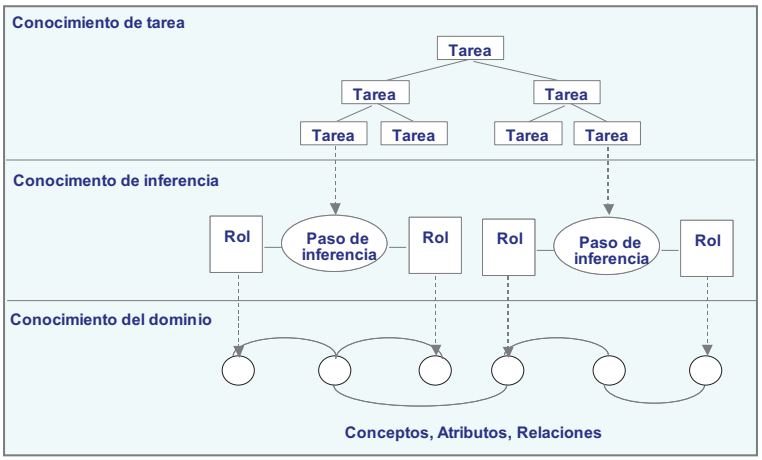


Figura 8.1: Categorías de conocimiento consideradas por CommonKADS.

El nivel del dominio consta de los hechos y suposiciones relevantes en el razonamiento para llevar a cabo una tarea en el dominio de aplicación. Se trata de un conocimiento expresado de forma declarativa, independiente de su uso. Consta de: (1) *esquema del dominio*: descripción esquemática del conocimiento específico en cuanto a definiciones de tipos o generalizaciones de estructuras del dominio y (2) *bases de conocimiento*: instancias de los tipos definidos en el esquema del dominio. Por ejemplo el siguiente conjunto de sentencias corresponde a sentencias sobre el esquema del dominio (ejemplos extraídos de [Schreiber et al., 00]):

```
CONCEPT fuel-tank;
ATTRIBUTES
status: {full, almost-empty, empty}
END CONCEPT fuel-tank;
```

```
BINARY-RELATION owned-by
INVERSE: owns;
ARGUMENT-1: car; CARDINALITY 0-1;
ARGUMENT-2: person; CARDINALITY: ANY;
ATTRIBUTES: purchase-date: DATE;
END BINARY-RELATION owned-by

RULE-TYPE state-dependency;
ANTECEDENT: invisible-car-state;
CONSEQUENT: car-state;
CONNECTION-SYMBOL: causes;
END RULE-TYPE state-dependency;
```

El siguiente conjunto de sentencias ilustra el caso de reglas de una base de conocimiento con el formato derivado de las sentencias anteriores:

```
fuel-supply.status = blocked
CAUSES gas-in-engine.status = false

battery.status = low
CAUSES  power.value = off
```

El conocimiento de inferencia en CommonKADS expresa los pasos de inferencia básicos que actúan sobre el dominio. Por ejemplo en la figura 8.2. el paso de inferencia de cubrir (*cover*) tiene un rol dinámico de entrada (*complaint*) y un rol dinámico de salida (*hypothesis*). En la metodología se relacionan los elementos del nivel de dominio que corresponden a dicha información mediante elementos de traducción entre nivel de inferencia y nivel del dominio (*inference-domain mapping*). Así, por ejemplo, según la figura, el concepto estado visible del vehículo

(*visible-car-state*) desempeña el rol de síntoma (*complaint*) en el paso de inferencia de cubrir (*cover*).

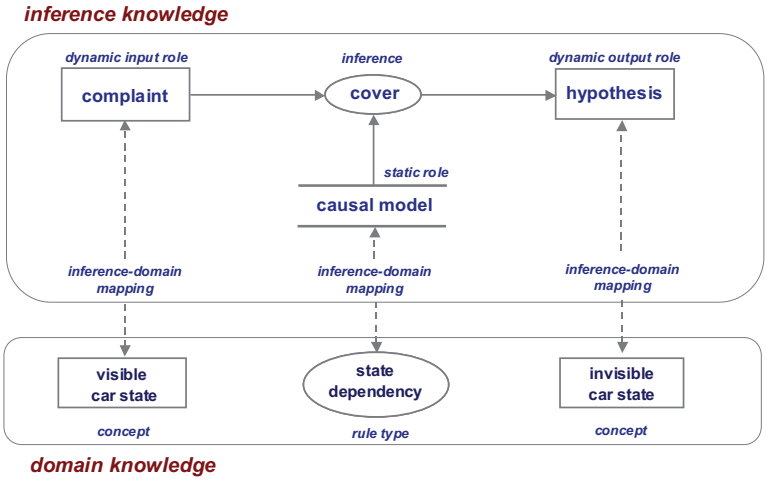


Figura 8.2: Relación entre nivel de inferencia y dominio (adaptación de figura publicada en [Schreiber et al., 00]).

Por otra parte, el conocimiento a nivel de tarea expresa cómo se reúnen los distintos pasos de inferencia para realizar tareas de alto nivel. Por ejemplo, la figura 8.3 muestra un caso expresado a nivel de tarea en donde una tarea (diagnóstico) se descompone mediante un método (generación y prueba) en un conjunto de subtareas entendidas como inferencias. Obsérvese que esta figura presenta similitudes con la forma de describir métodos de resolución de problemas mostrada en capítulos anteriores.

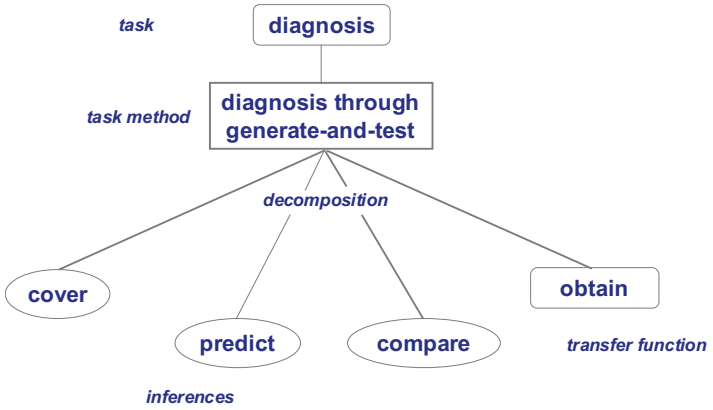


Figura 8.3: Descomposición de una tarea (adaptación de [Schreiber et al., 00]).

En resumen, la metodología CommonKADS aporta una solución interesante para análisis de conocimiento incorporando medios descriptivos para formular los métodos de resolución de problemas.

Tipo de Problema	***Métodos aplicables***
Diagnóstico	*Basado en consistencia*
	Basado en asociaciones
	Basado en dependencia causal
	.....
Predicción	*P. cualitativa comportam.*
	P. cualitativa de valor
	Simulación cualitativa
	.....
Evaluación	*Equiparación de casos*
	Especificación y equiparación
	Abstracción y equiparación
	.....
Diseño	*Diseño rutinario*
	Diseño innovativo
	Diseño original
	.....
Configuración	*Basado en casos*
	Orientado a recursos
	Orientado a estructura
	.....
Planificación	*Planificación lineal*
	Planificación no lineal
	Por patrones
	.....
Asignación	*Proponer y revisar*
	Proponer y revisar. jerárquico
	Descomposición múltiple
	.....
Modelización	*Basada en componentes*
	.....

Figura 8.4: Visión general de la biblioteca de métodos de CommonKADS

8.2 Bibliotecas de métodos

Uno de los recursos de ingeniería del conocimiento para ayuda a la construcción de sistemas inteligentes es una biblioteca de métodos de resolución de problemas que reúne un conjunto suficientemente amplio para diversas clases de problemas. Así, el desarrollador, cuando se enfrenta a la construcción de un nuevo sistema, puede buscar en dicha biblioteca el método (o métodos) que mejor encaja con el problema a resolver y, una vez seleccionado, el método sirve de guía para el proceso de adquisición del conocimiento además de plantilla de diseño de la aplicación informática. Las bibliotecas de métodos son una herramienta muy útil dado que permiten aprovechar la experiencia previa en la construcción de modelos.

Este tipo de bibliotecas incluyen:

- Un conjunto de métodos de resolución de problemas, cada uno descrito por:
  - la tarea con los tipos de argumentos (entradas y salidas)
  - los roles de conocimiento (tipos de conocimiento)
  - las subtareas o pasos de inferencia que lleva a cabo
  - la estrategia de inferencia (algoritmo)

- Conocimiento de selección de métodos para elegir el método más apropiado para un caso determinado. Esto, además de otros métodos de selección más sofisticados incluye al menos una clasificación de los métodos atendiendo a criterios tales como clases de problemas que resuelve.

Como ejemplos de bibliotecas se pueden citar los trabajos de Richard Benjamins para métodos de diagnóstico [Benjamins, 93] o la biblioteca de CommonKADS para diversas clases de problemas [Breuker, Van de Velde, 94]. Por ejemplo, en la figura 8.4 se muestra un resumen de los métodos de resolución de problemas considerados para diversas clases de problemas. Dicha metodología establece no

sólo un conjunto de problemas y métodos a considerar para cada clase de problema sino que, además, en cada clase de problema se definen una serie de criterios para determinar cuál es el método más adecuado. La figura 8.5 muestra un ejemplo de este conjunto de criterios en forma de árbol de decisión para el caso de problemas de diagnóstico. En la figura, partiendo del nodo superior, se realizan diversas preguntas cuya respuesta dirigen al desarrollador hacia el método más adecuado.

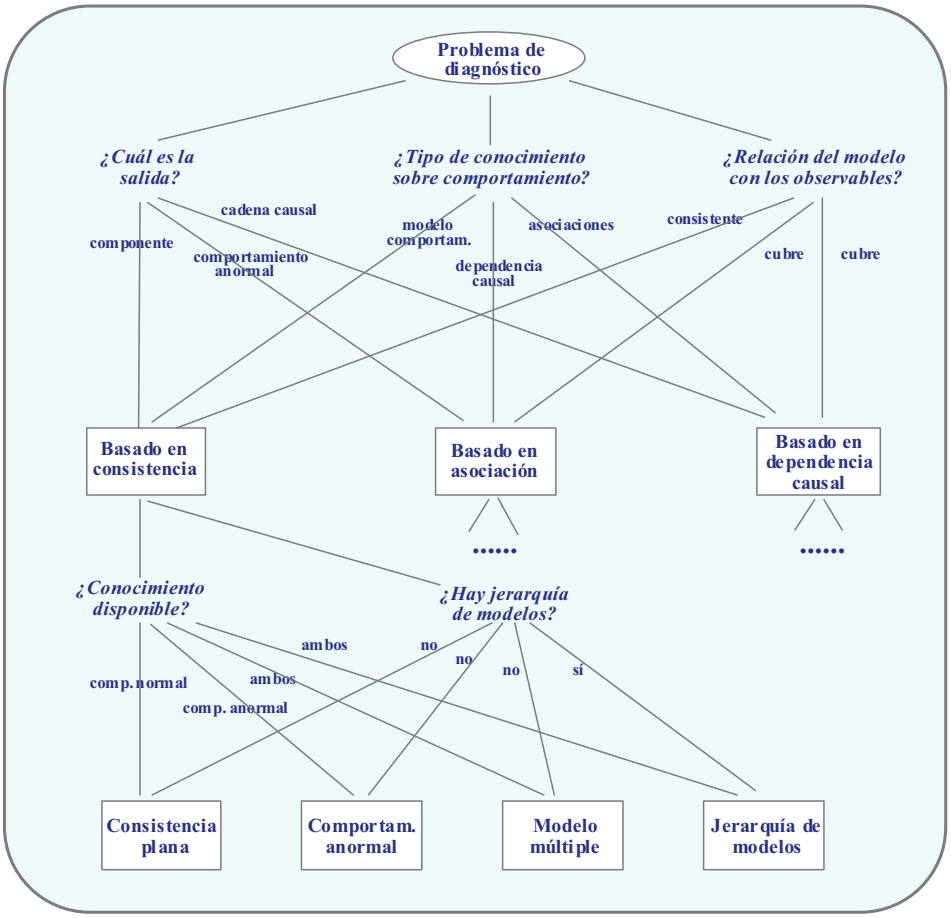


Figura 8.5: Ejemplo de criterios de selección de métodos en la biblioteca CommonKADS

8.3 Herramientas de ayuda a la construcción

Para ayudar a la construcción de sistemas inteligentes, el desarrollador cuenta con un amplio repertorio de herramientas que van desde las más sencillas como lenguajes de programación especializados a las más sofisticadas como los entornos de desarrollo para modelización. De una forma resumida, dichas herramientas se pueden agrupar de la siguiente forma (ordenadas de menor a mayor soporte en el desarrollo):

- *Lenguajes de programación especializados*. Se trata de lenguajes de programación que facilitan la construcción de sistemas inteligentes mediante el uso de representaciones simbólicas y/o uso de mecanismos de control alternativos. Por ejemplo los lenguajes Prolog (programación lógica), Lisp (programación funcional) y CLIPS (lenguaje basado en reglas de producción).

- *Herramientas de adquisición especializadas en una representación*. Estas herramientas (que se han denominado también herramientas *shell*) aportan la arquitectura del sistema inteligente (motor de inferencia, sistema explicativo, etc) pero tienen la base de conocimiento vacía. Para construir un sistema es necesario únicamente proporcionar el contenido de la base utilizando las facilidades que proporciona la herramienta como el lenguaje de representación y el mecanismo de validación. Las herramientas *shell* están normalmente asociadas a un único motor de inferencia con una representación uniforme del conocimiento. La idea original de este tipo de herramientas partió con el sistema EMYCIN [van Melle et al., 81] y fue seguida después con diferentes soluciones comerciales.

- *Herramientas de adquisición especializadas en un método*. Se trata de herramientas similares a las anteriores pero que están asociadas a un método

y, por tanto, a una clase de problemas. Debido a ello, pueden aportar más facilidades para ayudar a la construcción del sistema. Dentro de esta categoría, por ejemplo, están las herramientas SALT [Marcus, McDermott, 89] con el método proponer y revisar y MOLE [Eshelman, 88] con el método cubrir y diferenciar.

- *Entornos de modelización del conocimiento*. Son herramientas que no están comprometidas con ninguna representación ni método y que están orientados a facilitar el proceso de construcción de modelos de forma general.

8.3.1. Los entornos de modelización del conocimiento

Los entornos de modelización tienen dos fines principales: (1) ayudar al desarrollador en la aplicación de una metodología para análisis del conocimiento o bien (2) ayudar al desarrollador y al usuario final respectivamente en la construcción y mantenimiento del sistema basado en el conocimiento. Dentro del primer grupo se pueden citar herramientas que ayudan a realizar el análisis del conocimiento de acuerdo a metodologías basadas en CommonKADS tales como Shelley o Mike [Angele et al. 92] que hace uso de un lenguaje lógico KARL para construir prototipos de modelos de conocimiento. Por otro lado, dentro de la segunda categoría se encuentran las herramientas KREST de Universidad libre de Bruselas [Steels, 92], Protégé-II de Universidad de Stanford [Puerta et al., 93] y KSM de Universidad Politécnica de Madrid [Molina, 93; Cuena, Molina, 97; Cuena, Molina, 00] que ayudan a la construcción del sistema final haciendo uso de componentes software basados en el conocimiento de alto nivel.

En particular KSM (*Knowledge Structure Manager*) es un entorno software que permite la formulación de modelos abstractos de conocimiento haciendo uso de un lenguaje descriptivo parcialmente inspirado parcialmente en la metodología

CommonKADS y extendido con entidades conceptuales integradoras. Dichos modelos abstractos se asocian con componente software que permiten su traducción directa a la versión ejecutable. Los modelos abstractos de conocimiento se particularizan en dominios concretos haciendo uso de los lenguajes declarativos de las bases de conocimiento, lo cual aporta un elevado índice de generalidad y flexibilidad.

En concreto, en el entorno KSM un modelo de conocimiento se expresa haciendo uso del concepto básico denominado área de conocimiento mediante la cual se puede describir una jerarquía de áreas (con relaciones de agregación) para indicar cómo el modelo general se descompone en partes. Cada área, internamente, se describe mediante (1) el conocimiento local, expresado con otras áreas y (2) los problemas que puede resolver, expresado como una colección de tareas. El paradigma de modelización del conocimiento que se propone con KSM está formado por tres perspectivas fundamentales: la perspectiva de áreas de conocimiento, la perspectiva de tareas y la perspectiva de vocabularios. En los próximos apartados se presentan cada una de ellas.

La perspectiva de *áreas de conocimiento* define la estructura central del modelo ofreciendo una visión global del conocimiento el sistema. Se trata de una primera imagen general que se complementará después con otras perspectivas de mayor detalle. Esta visión está basada en una concepción modular del conocimiento utilizando como componente central el área de conocimiento. Un *área de conocimiento* identifica un cuerpo de conocimiento que explica un determinado comportamiento de resolución de problemas del agente que se modeliza. Ejemplos de áreas de conocimiento son: la comprensión del funcionamiento de un reactor nuclear o la gestión del tráfico de un área urbana. Habitualmente un área de conocimiento identifica una aptitud profesional, una cualificación o una especialidad de uno o varios expertos. La naturaleza del área de conocimiento se concibe como la naturaleza del conocimiento entendida de una forma

antropomórfica (tal como indica [Newell, 82]), no como una colección de procesos o estructuras de datos que corresponde más a una visión de proceso de información.

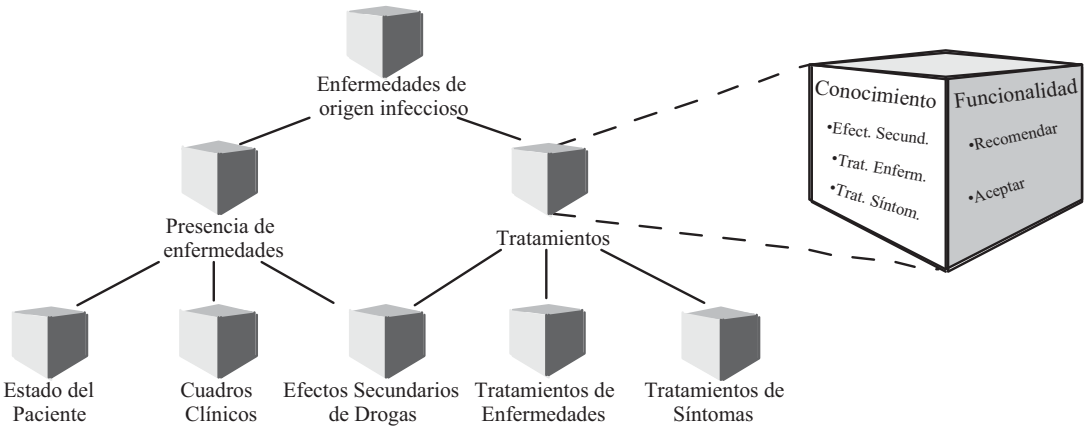


Figura 8.6: Ejemplo de perspectiva de áreas de conocimiento de un modelo de conocimiento

Para describir el modelo de conocimiento de un determinado agente, un desarrollador identifica un conjunto de áreas de conocimiento de forma que el modelo completo se contempla como una colección estructurada de dichas áreas. El modelo global se representa mediante un área de conocimiento de alto nivel. Por ejemplo, podría haber un área denominada enfermedades de origen infeccioso que representa el conocimiento completo sobre este dominio (figura 8.6). El área se refina identificando otras áreas de mayor detalle incluídas en la primera. Así, el área de enfermedades infecciosas se podría dividir, por un lado, en un área sobre presencia de enfermedades que permite detectar la presencia y causa de dolencias y, por otro lado, en un área sobre tratamientos que establece terapias para determinadas enfermedades. A su vez, cada área puede refinarse de nuevo dividiéndose en otras áreas más simples progresivamente desarrollando una estructura jerárquica (la estructura generada admite que ciertas áreas formen parte de más de un área de nivel superior). Al final de este refino se encuentran áreas primarias, es decir, áreas que no se descomponen en otras de más detalle. El

concepto de área primaria es una decisión subjetiva del desarrollador que establece cuando considera que puede ser directamente modelizada por una técnica básica de representación del conocimiento. Internamente un área de conocimiento particular se define con una doble visión: (1) conocimiento, es decir, el conjunto de áreas componentes y (2) funcionalidad, es decir, el conjunto de tareas o acciones que el área puede suministrar. La estructura completa identifica un patrón conceptual como una colección organizada de cuerpos de conocimiento definiendo la forma del modelo pero no su contenido real. Será en etapas posteriores durante la implementación del sistema cuando el desarrollador realice una adquisición del conocimiento detallada para construir las bases de conocimiento particulares.

La estructura de áreas de conocimiento ofrece una visión declarativa de la organización del conocimiento mostrando de forma modular cómo unos cuerpos de conocimiento están incluidos en otros, además del tipo de problemas que el modelo es capaz de resolver. Pero esta visión no presenta explícitamente los diferentes tipos de razonamiento que se realizan para resolver problemas. Para ello se utiliza lo que se denomina la *perspectiva de tareas*. Así, una *tarea* es una acción de resolución de problemas soportada por un área de conocimiento. La tarea recibe un conjunto de datos de entrada y genera un conjunto de datos de salida como resultado de su razonamiento. Ejemplos de tareas son: el diagnóstico médico de un paciente, la asignación de un conjunto de oficinas a un grupo de personas o el diseño del sistema mecánico de un ascensor. La tarea se contempla como uno de los múltiples comportamientos a los que puede dar lugar un área de conocimiento. Por ejemplo, considérese el área de enfermedades infecciosas. Saber de este campo significa ser capaz de realizar diagnóstico sobre esta clase de enfermedades y, además, poder proporcionar una terapia adecuada. También permite responder si una cierta medicina propuesta por otro especialista puede interaccionar negativamente con una enfermedad infecciosa existente. Estas tres funciones (diagnóstico, proponer tratamiento y aceptar una medicina propuesta) son tareas inherentes al área de conocimiento sobre enfermedades infecciosas. Por tanto, un modelo que dispone de

una perspectiva de áreas de conocimiento admite varias perspectivas de tareas, una por cada tarea global que el sistema puede ofrecer.

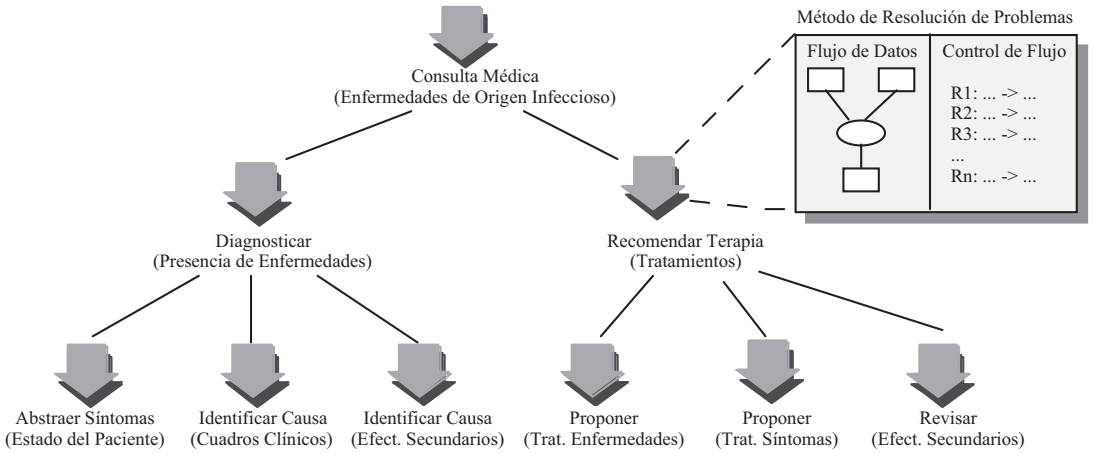


Figura 8.7: Ejemplo de perspectiva de tareas de un modelo de conocimiento.

Una tarea se lleva a cabo mediante un *método de resolución de problemas*. El método establece una estrategia de razonamiento manejando un conjunto de subtareas. Por ejemplo, la tarea de diseño del sistema mecánico de un ascensor puede llevarse a cabo por el método de proponer-y-revisar, el cual identifica tres subtareas: proponer diseños, revisar restricciones y solucionar incompatibilidades. Así, el uso de un determinado método para realizar una tarea descompone ésta en tareas más sencillas. La continuación de este proceso de modelización desarrolla un árbol de tareas-subtareas dando lugar a la perspectiva completa de tareas. Las tareas de las hojas del árbol corresponden a tareas asociadas a áreas de conocimiento elementales (figura 8.7).

Cuando se proponen medios descriptivos que organizan el conocimiento de forma modular pueden existir dependencias entre los componentes si su conocimiento se refiere a conceptos comunes. Por ejemplo, considérese una aplicación que supervisa el funcionamiento de una instalación industrial. El conocimiento de la aplicación

puede estar distribuído en un módulo que razona sobre detección de fallos y otro sobre propuestas de acciones de control para solucionar los fallos. Estos dos módulos comparten una visión de la instalación formada por los elementos físicos sobre los que razonan tales como los evaporadores, las válvulas, los conductos, etc., que representan la configuración física de la instalación. Cada módulo incluye su propio conocimiento sobre el área local que representa, pero los dos comparten la visión de la instalación. Para ello, es necesario manejar una terminología común que asegure que se produce un entendimiento completo entre los diversos módulos de conocimiento que forman parte del modelo. De acuerdo con esta idea, en KSM se utiliza lo que se denomina vocabulario conceptual. Un vocabulario define un conjunto de conceptos para establecer una terminología común sobre las que se define el conocimiento local de cada módulo que comparte el vocabulario. Así, un vocabulario no es local a una unidad sino que es como un idioma global compartido entre varias unidades. En general, un modelo completo dispondrá de varios vocabularios asociados a áreas de conocimiento primarias.

8.3.2 Implementación de modelos con KSM

Tras la etapa de análisis que da lugar a un modelo a nivel del conocimiento es necesario llevar a cabo la instrumentación de dicho modelo con objeto de obtener una versión operativa que resuelva automáticamente problemas en el ordenador. Para construir esta versión es necesario asociar a los diversos componentes del modelo diferentes elementos computables. Para ello, KSM proporciona tres herramientas básicas: las primitivas de representación para construir áreas de conocimiento primarias, el lenguaje Link para formalizar y ejecutar métodos de resolución de problemas y el lenguaje Concel para formalizar vocabularios conceptuales. Se trata de tres componentes de alto nivel que presentan importantes analogías con el concepto de modelización mostrado en la sección anterior. Ello permite que el paso del análisis del problema y la descripción del conocimiento en términos formales a la construcción del modelo operativo en el ordenador sea

mucho más sencillo que una traducción a un lenguaje de programación convencional. El manejo de estos componentes permite contemplar la construcción del modelo final como un ensamblaje y adaptación de bloques reutilizables desde una perspectiva cognitiva que sustituye a concepto tradicional de programación.

La estructura central del modelo a nivel del conocimiento en KSM se define con una jerarquía de áreas de conocimiento, en donde cada área se divide en áreas más simples hasta alcanzar áreas elementales denominadas áreas primarias. Para construir la versión operativa de estas áreas en KSM se dispone de lo que se denominan primitivas de representación. Una *primitiva de representación* es un componente reutilizable de software que implementa una técnica genérica de resolución de clases de problemas. Por ejemplo, una técnica muy extendida para representación y resolución de problemas es el modelo de reglas de producción que habitualmente está soportado por herramientas (los shells) para construcción de bases de conocimiento.

Otros ejemplos de primitivas son: representación en marcos con procedimientos de equiparación, representación en restricciones con procedimientos de satisfacción, representación en lógica con procedimientos de demostración automática, etc. Además de estas primitivas que utilizan técnicas basadas en el conocimiento se pueden considerar también otras que manejan técnicas convencionales como son: primitiva de manejo de series temporales con procedimientos de predicción, agregación, etc., primitiva de grafos con procedimiento de búsqueda de caminos, etc. La primitiva de representación está dividida en dos componentes principales. Por un lado, una representación declarativa del conocimiento (por ejemplo, la primitiva basada en reglas maneja hechos con formato concepto-atributo-valor y reglas si-entonces que los relacionan de forma condicional) y, por otro lado, un conjunto de tareas que muestran la clase de problemas que la primitiva puede resolver (por ejemplo la primitiva basada en reglas tiene la tarea denominada

encadenamiento deductivo que determina si una meta puede deducirse a partir de un conjunto de premisas utilizando las reglas de la base de conocimiento).

```
METODO establecer y refinar
ARGUMENTOS
       ENTRADA descripción
       SALIDA categoría
FLUJO DE DATOS
       (validez) establecer
              ENTRADA descripción, hipótesis
              SALIDA categoría
       (taxonomía) refinar
              ENTRADA hipótesis
              SALIDA hipótesis
CONTROL DE FLUJO
       INICIO
       -> (taxonomía) refinar,
          (validez) establecer.

       (validez) establecer ES establecida,
       (taxonomía) refinar ES hipótesis intermedia
       -> (taxonomía) refinar,
          (validez) establecer.

       (validez) confirmar ES establecida,
       (taxonomía) refinar ES hipótesis final
       -> FIN.
```

Figura 8.8: Ejemplo de formulación de un método de resolución de problemas utilizando el lenguaje Link

En KSM se maneja una biblioteca de primitivas de representación que incluye las más habituales en la construcción de sistemas. Para construir un modelo particular, el desarrollador elige en cada momento la primitiva más adecuada para cada área de conocimiento primaria y la adapta en el dominio del modelo. Esta adaptación es un proceso de construcción de una base de conocimiento para una determinada área de conocimiento que supone manejar la representación que aporta la primitiva. No obstante, la biblioteca de primitivas está concebida de forma abierta. Así, en el

proceso de construcción del modelo puede no existir una primitiva concreta para modelizar una determinada área de conocimiento. En ese caso el desarrollador debe programar una nueva primitiva utilizando un lenguaje de programación convencional que implemente la técnica necesaria y, a continuación, la nueva primitiva se incluye en la biblioteca para ser utilizada en diferentes aplicaciones. La biblioteca de KSM que inicialmente contiene un conjunto de primitivas generales puede crecer progresivamente recogiendo nuevas técnicas de representación para ser reutilizadas en la construcción de nuevas aplicaciones. Esta concepción abierta de la biblioteca evita el compromiso con una determinada representación del conocimiento permitiendo al desarrollador elegir la representación más conveniente en cada caso.

En la definición del modelo de conocimiento, el desarrollador indica cómo llevar a cabo cada tarea asociando un método de resolución de problemas que define una determinada estrategia de razonamiento con un uso particular de conocimiento.

En KSM para formular y ejecutar métodos se utiliza un lenguaje especialmente diseñado para este propósito que recibe el nombre de Link. El lenguaje Link establece para un determinado método una estrategia de razonamiento que indica cómo utilizar un conjunto de subtareas. Fundamentalmente, el método se describe con tres partes: (1) argumentos, es decir, las entradas y salidas del método, (2) el flujo de datos, que establece la conexión entre subtareas y (3) el control de flujo, que establece el orden de ejecución de las subtareas. Una de las contribuciones originales de KSM es la representación del control de flujo. La representación propuesta, en vez de ser la tradicional formulación algorítmica de los lenguajes convencionales de programación con mecanismos de control de bucles, secuencias, repeticiones, etc., considera como no deterministas los pasos de razonamiento con lo que la secuencia de ejecución puede desarrollar un proceso de búsqueda en vez de una secuencia lineal de ejecución. En particular, Link maneja reglas sencillas de producción para definir este control. El formato de las reglas incluye (1) en la parte

izquierda de las reglas un conjunto de condiciones sobre estados intermedios de ejecución sobre el grado de éxito de ejecuciones previas de tareas, y (2) en la parte derecha de las reglas se incluye una secuencia de ejecución de tareas indicando modos de ejecución como por ejemplo umbrales de equiparación, máximo número de respuestas, time-out, etc. KSM incluye un intérprete de lenguaje Link que lleva a cabo la ejecución de los métodos. El intérprete realiza un proceso de encadenamiento hacia adelante que desarrolla un espacio de búsqueda local al método. Las llamadas a las subtareas pueden provocar la creación de otros espacios de búsqueda locales a dichas subtareas de forma que la ejecución completa da lugar a un árbol en donde cada nodo es un espacio de búsqueda local a un método. La figura 8.8 muestra un ejemplo completo de formulación de método en lenguaje Link que incluye la estrategia de establecer-y-refinar para llevar a cabo la tarea de clasificación (obsérvese que en la parte de control de flujo bastan tres reglas para definir la estrategia de razonamiento del método; en general será preciso escribir conjuntos pequeños de reglas para definir estrategias de razonamiento).

```
CONCEPTO tramo urbano SUBCLASE DE tramo.
ATRIBUTOS:
       Capacidad (INTERVALO RANGO 0 2000) [Veh_Km],
       Carriles (ENTERO RANGO 1 4): 1,
       Detectores (INSTANCIAS DE Detector),
       Longitud (ENTERO RANGO 0 1000) [m],
       Intensidad {alta, baja, media}
       Velocidad {alta, baja, media}
       Régimen {libre, saturación, congestión}.

CONCEPTO calle principal ES UN tramo urbano.
ATRIBUTOS:
       Capacidad: [1400, 1800] [Veh_Km],
       Carriles: 3,
       Detectores: (D1003, D1005),
       Longitud: 350 [m].
```

Figura 8.9: Ejemplo parcial de formulación de un vocabulario conceptual utilizando el lenguaje Concel

Finalmente, otra herramienta de modelización que suministra el entorno KSM es el lenguaje Concel que permite formular los vocabularios conceptuales identificados en el modelo. Este lenguaje utiliza una representación que permite definir conceptos, atributos, facetas y su clasificación en clases e instancias. Por ejemplo, la figura 8.9 muestra un ejemplo de parte de un vocabulario conceptual utilizando Concel, en donde se define en primer lugar un concepto clase, el tramo urbano, con un conjunto determinado de atributos y, a continuación, se define un concepto instancia de la clase anterior con valores concretos para dichos atributos.

8.3.3 El entorno KSM

KSM es un entorno informático que asiste a desarrolladores y usuarios finales en la construcción y mantenimiento de aplicaciones complejas que utilizan tanto técnicas basadas en el conocimiento como técnicas convencionales. El entorno suministra por un lado una interfaz gráfica de usuario para asistir en el proceso de construcción y ejecución de modelos y, por otro lado, el entorno suministra una biblioteca de primitivas de representación programadas con los lenguajes C++ y Prolog. El entorno asiste al desarrollador en diferentes etapas del ciclo de vida de una aplicación: análisis, diseño, implementación y mantenimiento.

El desarrollador durante la etapa de *análisis* utiliza el paradigma de modelización de KSM para definir un modelo conceptual que ha de ser aceptado por el usuario final antes de comenzar su implementación. A diferencia de los modelos tradicionales de ingeniería del software basados en una perspectiva de proceso de información, en KSM se utiliza una visión cognitiva que proporciona una imagen más rica e intuitiva de la arquitectura de la aplicación y que permite integrar aplicaciones de tipo convencional con aplicaciones basadas en el conocimiento.

En el *diseño* e *implementación* KSM proporciona un soporte para construir la versión ejecutable del modelo. Para ello KSM maneja una biblioteca abierta de

componentes de software reutilizables (las primitivas de representación) que el desarrollador adapta y ensambla siguiendo la estructura del modelo conceptual (utilizando el lenguaje Link). Con KSM se entiende la arquitectura de la aplicación con una estructura modular de dichos componentes en donde cada parte puede ser creada y validada individualmente antes de su integración en la arquitectura global.

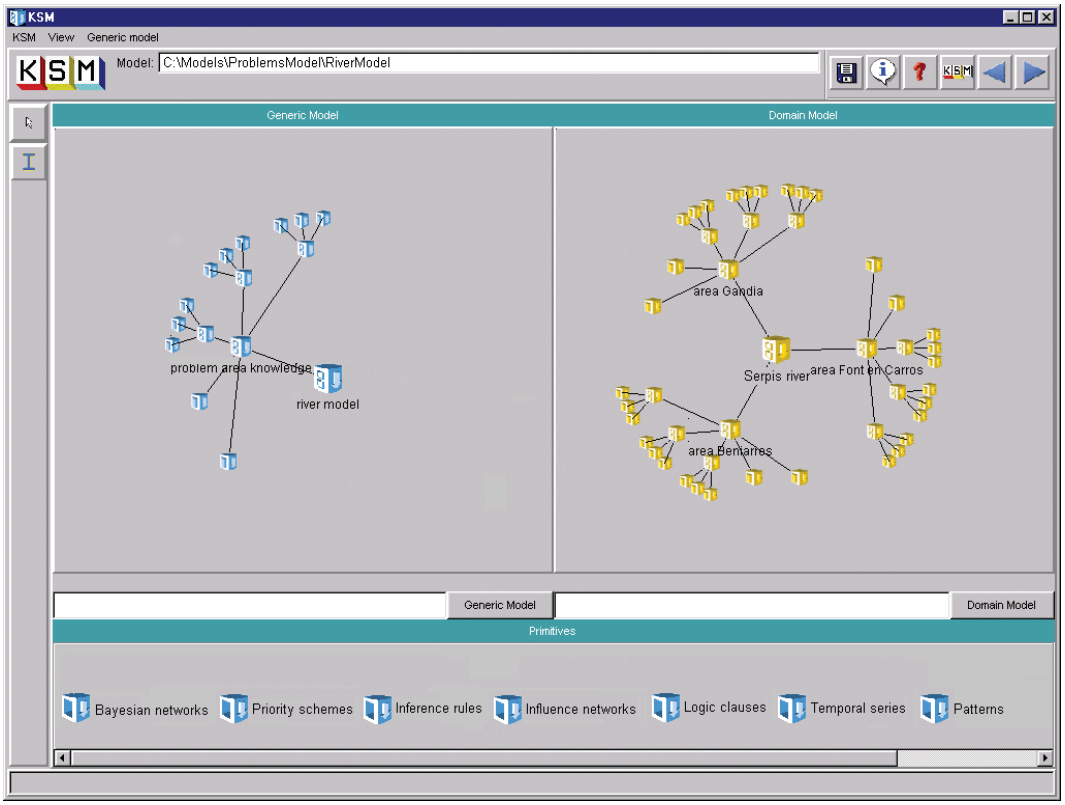


Figura 8.10: Ejemplo de pantalla principal mostrada por el entorno KSM

En las etapas de *operación* y *mantenimiento,* el usuario final puede utilizar KSM para consultar la estructura del modelo conceptual de la aplicación y puede acceder a bases de conocimiento locales siguiendo esta estructura. El papel de KSM en esta etapa es permitir al usuario final abrir la aplicación para acceder a su estructura de conocimiento de forma que, en vez de ser una caja negra como las aplicaciones

convencionales, la aplicación final presenta descripciones comprensibles de su conocimiento. El usuario además puede cambiar el modelo conceptual de la aplicación sin programar para adaptar el sistema de acuerdo con nuevos requisitos.

Una de las características más destacables del entorno KSM es su utilización como plataforma de soporte a la reutilización de modelos [Molina, Shahar, 96]. KSM maneja componentes abiertos de forma que el usuario puede acceder a su conocimiento utilizando una representación declarativa. El manejo de bases de conocimiento locales a las primitivas de representación permite que la misma primitiva puede ser utilizada para construir módulos distintos que incluyen diferentes bases de conocimiento, aunque tienen los mismos procedimientos de inferencia. El grado de reutilización dependerá de la riqueza expresiva que aporte la representación del conocimiento, yendo desde un nivel mínimo en donde no se maneja una representación explícita del conocimiento (todo el conocimiento se encuentra codificado en los algoritmos de los programas), hasta un grado máximo en donde se manejan representaciones tales como marcos, restricciones o reglas.

En KSM es posible definir además lo que se denomina *modelo genérico* que es una estructura abstracta de conocimiento formada por áreas genéricas. Así por ejemplo, se puede definir un modelo general sobre gestión de tráfico urbano que incluye de forma abstracta las distintas áreas de conocimiento que intervienen en este dominio como son, conocimiento sobre detección de accidentes, congestiones, o conocimiento sobre actuaciones (policía, cambios de estado de semáforos, etc.). Después, esta estructura puede utilizarse para definir el modelo particular de control de tráfico de la ciudad de Madrid, en donde se duplicarán algunas áreas concretas, pero la estructura esencial será la que indica el modelo genérico. Esta capacidad de manejo de modelos genérales es una herramienta muy útil en el proceso de ingeniería del conocimiento dado que permite reutilizar estructuras para diversos dominios.

Así, KSM proporciona dos niveles de reutilización. Por un lado, el manejo de componentes preprogramados que presentan un nivel mayor de reutilización al estar concebidos de forma abierta, lo cual simplifica la actividad de instrumentación de los modelos. Por otro lado, la posibilidad de reutilizar modelos generales que se pueden instanciar en diversos dominios, lo que puede facilitar significativamente la actividad de diseño de nuevos modelos.

En conclusión, la propuesta del entorno KSM supone un cambio importante respecto al tradicional concepto de construcción de aplicaciones. El desarrollador que utiliza KSM concibe el desarrollo de una aplicación como una actividad de formulación de un modelo del conocimiento que interviene en el sistema final. La construcción de dicho modelo se realiza, no mediante el enfoque clásico de programación de instrucciones, sino como una actividad de selección, adaptación y ensamblaje de componentes reutilizables de alto nivel con una formulación basada en el conocimiento, lo que al ser más intuitivo y natural da lugar a una mayor comprensión por usuarios y diseñadores no programadores.

Referencias

[AAAI, 04] Buscador de la AAAI: American Association for Artificial Intelligence: http://www.aaai.org/pathfinder/ 2004.

[Agosta, 95] Agosta J.M.: “Formulation and implementation of an equipment configuration problem with the SIPE-2 generative planner”. Proc. AAAI-95 Spring Symposium on Integrated Planning Applications, pp. 1-10, 1995.

[Alonso et al. 90] Alonso M., Cuena J., Molina M."SIRAH: An Architecture for a Professional Intelligence", ECAI Conference 1990. (L.Carlucci Ed.) Pitman, 1990.

[Angele et al., 92] Angele J., Fensel D., Landes D., Neubert S., Studer R.: "Model-based and Incremental Knowledge Engineering: The MIKE Approach". Proc. Artificial Ingelligence from the Information Processing (Workshop AIFIPP 92), Madrid, 1992.

[Ben-Bassat, 78] Ben-Bassat, M.: "Myopic Policies in Sequential Classification". IEEE Transactions on Computers. C-27, 170-178. 1978.

[Benjamins, 93] Benjamins, V.R.: "Problem Solving Methods for Diagnosis", PhD Thesis, University of Amsterdam, 1993.

[Breuker, Van de Velde, 94] Breuker J., Van de Velde W.: "CommonKADS Library for Expertise Modelling: Reusable Problem Solving Components". IOS Press. 1994.

[Brown, Chandrasekaran, 89] Brown D., Chandrasekaran B.: "Design Problem-solving: Knowledge Structures and Control Strategies", Morgan Kaufman, 1989.

[Boy, Gruber, 90] Boy, G., Gruber T.R.: "Intelligent Assistant Systems: Support for Integrated Human-Machine Systems" Technical Report KSL 90-61, Knowledge Systems Laboratory, Computer Science Department, Stanford University, 1990. También en proceedings of 1990 AAAI Spring Symposium on Knowledge-Based Human-Computer Communication, March 1990, Stanford University.

[Buchanan, Shortliffe, 84] Buchanan B.G., Shortliffe E.H.:"Rule-based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project". Addison Wesley. 1984.

[Bylander, 91] Bylander, T.: "Complexity Results for Planning". Proceedings 12th International Joint Conference on Artificial Intelligence IJCAI 91, Sydney, Australia, August, 1991.

[Bylander et al., 91] Bylander, T., Allemang, D., Tanner, M.C., Josephson, J.R.: "The Computational Complexity of Abduction". Artificial Intelligence, 49, 25-60.

[Chandrasekaran, 90] Chandrasekaran, B.:"Design Problem Solving: A Task Analysis". AI Magazine 11, 4 (1990), 59-71.

[Chandrasekaran, Mittal, 89] Chandrasekaran, B., Mittal S.:"Deep Versus Compiled Knowledge Approaches to Diagnostic Problem-solving". International Journal of Man-Machine Studies 19, 425-436. 1983.

[Chandrasekaran et al., 92] Chandrasekaran B., Johnson T.R, Smith J.W.: "Task Structure Analysis for Knowledge Modeling", Communications of the ACM, 35 (9), 124-137. 1992.

[Clancey, 84] Clancey W. J.: "Details of the Revised Therapy Algoritm". Capítulo en Buchanan B.G., Shortliffe E.H. (Eds) "Rule-based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project". Addison Wesley. 1984. También publicado en

en Proc. of the International Joint Conference on Artificial Intelligence IJCAI. W. Kaufmann, Los Altos California. 1977.

[Clancey, 85] Clancey W.: "Heuristic Classification". Artificial Intelligence 27, 1985

[Console, Torasso, 90] Console, L., Torasso, P.: "Hypotetical Reasoning in Causal Models". International Journal of Intelligent Systems, 5(1):83-124. 1990.

[Cuena, 98] Cuena, J.: "Sistemas Inteligentes: Conceptos, Técnicas y Métodos de Construcción". Fundación General de la Universidad Politécnica de Madrid. Servicio de Publicaciones de la Facultad de Informática. Universidad Politécnica de Madrid. 1998.

[Cuena, et al. 91] Cuena J., Molina M., Garrote L."An Architecture for Cooperation of Knowledge Bases and Quantitative Models: The CYRAH Environment", Second Generation Expert Systems, Avignon'91, EC2, 1991.

[Cuena, et al. 92] Cuena J., Garrote L., Molina M., "Combining Simulation Models and Knowled-ge Bases for Real Time Flood Management", Hydraulic Engineering Software HYDROSOFT'92. Computational Mechanics Publications. Wessex Institute of Technology, 1992.

[Cuena, Molina, 97] Cuena J., Molina M.: "KSM: An Environment for Design of Structured Knowledge Models". En

"Knowledge-based Systems: Advanced Concepts, Techniques and Applications". S. G. Tzafestas (Ed.). Publisher World Scientific Publishing Company. 1997.

[Cuena, Molina, 99] Cuena J., Molina M.: "A Multi-agent System for Emergency Management in Floods". En "Multiple Approaches to Intelligent Systems" Iman I., Kodratoff Y., El-Dessouki A., Ali M. (Eds.) Lecture Notes in Artificial Intelligence 1611, Springer. Proc. 12th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE-99, Cairo, Egypt, May/June, 1999.

[Cuena, Molina, 00] Cuena J., Molina M.: "The Role of Knowledge Modeling Techniques in Software Development: A General Approach based on a Knowledge Management Tool". International Journal of Human-Computer Studies. No. 52. pp 385-421. Academic Press, 2000.

[de Kleer, 86] de Kleer, J.: "An Assumption Based Truth Maintenance System". Artificial Intelligence 28. 1986.

[de Kleer, 90] de Kleer, J.: "Using Crude Probability Estimates to Guide Diagnosis". Artificial Intelligence 45, pp 381-392. 1990.

[de Kleer, Brown, 84] de Kleer, J., Brown, J.: "A Qualitative Physics Based on Confluences". Artificial Intelligence. 24:7-83. 1984.

[de Kleer, Williams, 87] de Kleer, J., Williams, B.C.: "Diagnosing multiple faults". Artificial Intelligence, Vol. 32. 97-130. 1987

[de Kleer, Williams, 89] de Kleer, J., Williams, B.C.: "Diagnosis with behavioural modes". Proc 11th IJCAI, pp. 1324-1330. Detroit, 1989.

[Descotte, Latombe, 85] Descotte Y., Latombe J.: "Making Compromises among Antagonistic Constraints in a Planner". Artificial Intelligence 27, 1985.

[Doyle, 79] Doyle J.: "A Truth Maintenance System". Artificial Intelligence 12, pp. 231-272, 1979.

[Dressler, 88] Dressler O.: "Extending the basic ATMS". Proceedings of ECAI-88, 535-540. 1988.

[Duda, Reboh, 84] Duda R. O., Reboh R: "AI and Decision Making: The PROSPECTOR Experience". En Reitman W. (ed.) Artificial Intelligence Applications for Business, pp 111-147. Norwood, NJ, Ablex Publishing, 1984.

[Erol et al., 94] Erol K, Hendler J., Nau D. S.: "UMCP: A sound and complete procedure for hierarchical task-network planning." In Proceedings of the International Conference on AI Planning Systems (AIPS), pp. 249-254, 1994.

[Eshelman, 88] Eshelman L.: "MOLE: A Knowledge Acquisition Tool for Cover-and-differenciate Systems". In Marcus, S. (Ed.) "Automating Knowledge Acquisition for Expert

Systems", pp 37-80. Boston: Kluwer Academic Puclishers, 1988.

[Fensel, 95] Fensel, D.: "Assumptions and Limitations of a Problem-Solving Method: A Case Study". Proceedings of the 9th Knowledge Acquisition for Knowledge-Based Systems Workshop. Banff Canada. 1995.

[Fensel, 00] Fensel D.: "Problem-Solving Methods: Understanding, Description, Development and Reuse". Lecture Notes in Artificial Intelligence 1791, subserie de Lecture Notes in Computer Science. Springer Verlag. 2000.

[Frainier et al., 94] Frainier R., Groleau N., Hazelton L., Colombano S., Compton M., Statler I., Szolovits P., Young L.: "PI-in-a-Box: A Knowledge-Based System for Space Science Experimentation". AI Magazine. Spring, 1994.

[Friedland, 79] Friedland P.E.: "Knowledge-based experiment design in molecular genetics". Proc. Sixth International Joint Conference on Artificial Intelligence, 285-287, Menlo Park, California. También: report STAN-CS-79-771, Stanford University. 1979.

[Forbus, de Kleer, 93] Forbus, K.D., de Kleer J.: "Building Problem Solvers". The Massachusetts Institute of Technology MIT Press. 1993

[Ghallab et al., 04] Ghallab M., Nau D., Traverso P.: "Automated Planning: Theory and Practice". Morgan Kaufmann. San Francisco, CA. 2004.

[Goel et al., 87] Goel A., Soundarajan N., Chandrasekaran B..: "Complexity in Classificatory Reasoning". Proceedings of 6th National Conference on Artificial Intelligence AAAI 87, Seattle, Washington, July 13-17, 1987.

[Hamscher, 91] Hamscher, W.: "Principles of Diagnosis: Current Trends and a Report on the First International Workshop". AI Magazine, pp 15-37.

[Hamscher et al., 92] Hamscher, W., Console, L., de Kleer, J.: "Readings in Model Based Diagnosis". Morgan Kauffmann. 1992.

[Hernández et al., 04] Hernández J., Molina M., Cuena J.: "Towards an Advanced HCI Through Knowledge Modelling Techniques" in "Knowledge Engineering and Agents Technologies" Cuena J., Demazeau Y., García-Serrano A., Treur J. (eds.) IOS Press, 2004 .

[Howard, 66] Howard R. A.: "Information value theory". IEEE Transactions on Systems Science and Cybernetics. Vol. SSC-2, 1, 22-26, agosto 1966.

[IJHCS, 96] International Journal of Human-Computer Studies, 44 (3-4), March/April 1996. Special Issue on Sisyphus II.

[Johnson, Soloway, 96] Johnson, L; Soloway, E.: "PROUST: Knowledge-based Program Understanding". IEEE Transactions on Software Engineering, 11(3): 267-275. 1985.

[Josephson et al., 96] Josephson J.R., Josephson S.G.: "Abductive Inference: Computation, Philosophy, Technology". Cambridge University Press, 1996.

[Kahn, 88] Kahn G.: "MORE: From Observing Knowledge Engineers to Automating Knowledge Acquisition". En Marcus S. (Ed.) "Automating Knowledge Acquisition for Expert Systems", pp 7-35. Boston, Kluwer Academic Publishers, 1988.

[Lindsay et al., 80] Lindsay R K,, Buchanan B. G. Feigenbaum, E A, Lederberg J.: "Applications of Artificial Intelligence for Organic Chemistry: The DENDRAL Project" New York, McGraw-Hill. 1980.

[Marcus et al., 87] Marcus S., Stout J., McDermott J.: "VT: An Expert Elevator Designer that uses Knowledge-based Backtracking". AI Magazine, Winter 1987.

[Marcus, McDermott, 89] Marcus S., McDermott J.: "SALT: A Knowledge Acquisition Language for Propose-and-Revise Systems". Artificial Intelligence, 39(1): 1-38. 1989.

[Mathlab Group, 77] Mathlab Group: "MACSYMA Reference Manual " (Technical Report) Computer Science Laboratory, MIT. 1977.

[McDermott, 88] McDermott J.: “Preliminary Steps Toward a Taxonomy of Problem Solving Methods”. In "Automating Knowledge Acquisition for Expert Systems", S. Marcus (ed.), Kluwer Academic, Boston, 1988.

[Miller et al., 82] Miller R.A., Pople H.E., Myers J.D.: “INTERNIST-1: An Experimental Computer-based Diagnostic Consultant for General Internal Medicine.” New English Journal Medicine, 307(8):468-476, 1982.

[Mittal et al., 79] Mittal S., Chandrasekaran B., Smith J.: “Overview of MDX: a System for Medical Diagnosis”. Proc. Third Symposium Computer Applications in Medical Care, Washington D.C., October 1979, pp 34-46.

[Mittal et. al, 86] Mittal S., Dym C., Morjaria M..: “PRIDE: An Expert System for the Design of Paper Handling Systems”. IEEE Computer 19 (7): 102-114. 1986.

[Molina, 93] Molina M.: "Desarrollo de aplicaciones a nivel cognitivo mediante entornos de conocimiento estructurado". Tesis Doctoral. Facultad de Informática de la Universidad Politécnica de Madrid. 1993.

[Molina, 01a] Molina M.: "Modeling Commercial Knowledge to Develop Advanced Agent-based Marketplaces for E-commerce". Fifth International Workshop CIA-2001 on Cooperative Information Agents. Artículo en libro: "Cooperative Information Agents". Lecture Notes in

Artificial Intelligence, nº 2182, Springer-Verlag.. Modena, Italia. Septiembre, 2001.

[Molina, 01b] Molina M.: "An Intelligent Sales Assistant for Configurable Products". First Asia-Pacific Conference on Web Intelligence, WI 2001. Artículo en libro: "Web Intelligence: Research and Development". Lecture Notes in Artificial Intelligence, nº 2198, Springer-Verlag. Maebashi, Japón. Octubre 2001.

[Molina, 05] Molina M.: "Intelligent Assistant for Public Transport Management". International Conference on Intelligent Computing, ICIC 05, Artículo en libro de actas del congreso publicado en Lecture Notes in Artificial Intelligence, Springer-Verlag. Hefei, China, Agosto 2005.

[Molina, Blasco, 03] Molina M., Blasco G.: "A Multi-agent System for Emergency Decision Support". Proc. Fourth International Conference on Intelligent Data Engineering and Automated Learning, IDEAL 03. Lecture Notes in Computer Science. Springer. Hong Kong, 2003.

[Molina et al., 98] Molina M., Hernández J., Cuena J.: "A Structure of Problem-solving Methods for Real-time Decision Support in Traffic Control". Journal of Human and Computer Studies (Academic Press) N.49, 577-600, 1998.

[Molina, Ossowski, 99] Molina M., Ossowski S.: “Knowledge Modelling in Multiagent Systems: The Case of the Management of a National Network” in “Intelligence in Services and Networks, Paving the Way for an Open Service Market”. Zuidweg H., Campolargo M., Delgado J., Mullery A. (eds.), pp. 501-513. Lecture Notes in Computer Science 1597, Springer.

[Molina, Robledo, 01] Molina M., Robledo M.: "A Knowledge Model for Automatic Configuration of Traffic Messages". In “Engineering of Intelligent Systems” Monostori L., Vancza J., Ali M. (Eds.) Lecture Notes in Artificial Intelligence 2070, Springer. Proc. 14th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE-01, Budapest, Hungary, June 2001.

[Molina, Shahar, 96] Molina M., Shahar Y.: "Problem-solving Method Reuse and Assembly: From Clinical Monitoring to Traffic Control". Proc. of Knowledge Acquisition for Knowledge Based Systems Workshop, KAW-96. Banff, Canada, 1996.

[Moses, 67] Moses J.: “Symbolic Integration”. Doctoral Dissertation, MIT. 1967.

[Motta, Zdrahal, 96] Motta E.,Zdrahal, Z.: “Parametric Design Problem Solving”. Proceedings of the 10th Knowledge Acquisition for Knowledge-Based Systems Workshop. Banff Canada. 1996.

[Nau et al., 03] Nau D.S., Au T.C., Ilghami O., Kuter U., Murdock W., Wu D., Yaman F.: "SHOP2: An HTN planning system". Journal of Artificial Intelligence Research, 20:379-404, 2003.

[Nebel, 96] Nebel B.: "Artificial Intelligence: A Computational Perspective" In "Principles of Knowledge Representation" Brewka G. (Ed.) CSLI Publicaions, Studies in Logic, Language and Information, Stanford, 1996.

[Newell, 82] Newell A.: "The Knowledge Level" In Artificial Intelligence Vol 18 pp. 87-127, 1982.

[Openclinical, 04] OpenClinical: Knowledge Management for Medical Care. AI Systems in Clinical Practice. http://www.openclinical.org/aisinpractice.html, 2004

[Parres, 98] Parres y García I. C.: "Propuesta de un modelo estructurado de conocimiento para el problema de armonización musical". Trabajo Fin de Carrera. Facultad de Informática. Universidad Politécnica de Madrid. Julio, 1998.

[Poeck 90] Poeck, K.: "Conception and Implementation of a Problem-specific Expert System Tool for Assignement with the Propose-and-Exchange Strategy" (in German), Diploma Thesis, University of Karlsruhe. 1990.

[Pople, 77] Pople H. E.: "The Formation of Composite Hypothesis in Diagnostic Problem Solving: An Exercise in Synthetic Reasoning". Proceedings of the Fifth International Joint Conference on Artificial Intelligence, pp. 1030-1037. Los Altos, California. William Kaufmann. 1977.

[Pople, 81] Pople H. E.: "Heuristic Methods for Imposing Structure on Ill-Structured Problems: The Structuring of Medical Diagnosis" in "Artificial Intelligence in Medicine" P. Szolovitz (ed.), Westview Press, 1981.

[Pople, 82] Pople H. E.: "Heuristic Methods for Imposing Structure on Ill-structured Problems: The Structuring of Medical Diagnosis". En Solovits, P. (Ed.) "Artificial Intelligence in Medicine" AAAS Selected Symposium 51, pp. 119-190. Boulder, CO. Westview Press, 1982.

[Pople et al., 75] Pople H. E., Myers J.D., Miller R.A.: "DIALOG: A Model of Diagnostic Logic for Internal Medicine". Advanced Papers of the Fourth International Joint Conference on Artificial Intelligence, pp 848-855. Los Altos, California. William Kauffman. 1975

[Puerta et al., 93] Puerta A., Tu S.W., Musen M.A.: "Modeling Tasks with Mechanisms". International Journal of Intelligent Systems, Vol 8, 1993.

[Puppe, 93] Puppe F.: "Systematic Introduction to Expert Systems: Knowledge Representations and Problem-solving Methods". Springer Verlag. 1993.

[Rich, 81] Rich, C.: “A Formal Representation for Plans in the Programmers’s Apprentice”. Proc. Seventh International Joint Conference on Artificial Intelligence. Menlo Park, California. 1044-1052. 1981.

[Sacerdoti, 75] Sacerdoti, E.: “The nonlinear nature of plans”. Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), pp. 206-214, 1975.

[Schreiber et al., 93] Schreiber G., Wielinga B.J, Breuker J.A. (eds): "KADS: A principled approach to knowledge-based system development". Academic Press. 1993.

[Schreiber et al., 00] Schreiber G., Akkermans H., Anjewierden A., De Hoog R., Shadbolt N., Van de Velde W., Wielinga B.: “Knowledge engineering and management. The CommonKADS methodology” MIT Press, 2000.

[Serna, 99] Serna R.: “Desarrollo de un sistema basado en el conocimiento para soporte a la venta de productos configurables por internet”. Trabajo Fin de Carrera. Facultad de Informática, Universidad Politécnica de Madrid. 1999.

[Steels, 84] Steels L.: “Second Generation Expert Systems”. Conference on Future Generation Computer Systems, Rotterdam. In Journal of Future Generation Computer Systems, 1 (4, June, 1985).

[Steels, 90] Steels L.: "Components of Expertise" AI Magazine, Vol 11 (2), 29-49, 1990.

[Steels, 92] Steels L.: "Reusability and Configuration of Applications by Non-programmers". Proc. Artificial Intelligence form the Information Processing Perspective (Workshop AIFIPP 92). Madrid, 1992.

[Stefik 93] Stefik M.: "Introduction to Knowledge Systems". Morgan Kaufmann. 1993.

[Sticklen et al. 89] Sticklen, J., Chandrasekaran, B., Bond, W.: "Applying a Functional Approach for Model-based Reasoning". Proc. of Workshop on Model Based Reasoning IJCAI, pp 165-176. Detroit. 1989.

[Struss, Dressler, 89] Struss, P., Dressler, O.: "Physical Negation – Integration Fault Models into the General Diagnostic Engine". Proc. 11th IJCAI, pp. 1318-1323. Detroit. 1989.

[Tate, 77] Tate A.: "Generating project networks". Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), pp. 888-893, 1977.

[Thos, 01] Thos Martín J..: "Diseño de un asistente inteligente para ayuda a la armonización musical". Trabajo Fin de Carrera. Facultad de Informática. Universidad Politécnica de Madrid. Julio, 2001.

[van Melle et al., 81] van Melle, W., Scott, A., Benett, J., Pearis, M.: "The EMYCIN Manual". Technical Report, Stanford University. Heuristic Programming Project. 1981.

[Wilkins, 88] Wilkins, D: "Practical Planning: Extending the Classical AI Planning Paradigm". Morgan Kauffmann, San Mateo, CA. 1988.

[Wilkins, desJardins, 01] Wilkins D., desJardins M.: "A call for knowledge-based planning". AI Magazine, 22 (1): 99-115, 2001.

[Yost, 92] Yost, G. R.: "Configuring Elevator Systems". Technical Report, Digital Equipment Corporation, 111 Locke Drive, Marlboro, USA. 1992.

[Zdrahal, Motta, 95] Zdrahal, Z., Motta E.: "An In-depth Analysis of Propose & Revise Problem Solving". Proceedings of the 9th Knowledge Acquisition for Knowledge-Based Systems Workshop. Banff Canada. 1995.

[Zdrahal, Motta, 96] Zdrahal, Z., Motta E.: "Improving Competence by Integrating Case-based Reasoning in Heuristic Search". Proceedings of the 10th Knowledge Acquisition for Knowledge-Based Systems Workshop. Banff Canada. 1996.

ANEXO A:
Notación para representación simbólica

En este apartado se describe la notación utilizada para representación simbólica en el texto. Principalmente se utilizan las representaciones básicas que incluyen hechos en forma de concepto-atributo-valor con organización en clases, reglas y restricciones (aunque algunos casos se utiliza también la notación en forma de cláusulas lógicas similar a la utilizada en programación lógica). Seguidamente se describe brevemente la notación utilizada en en estos casos.

A.1. Representación de hechos

Los hechos se representan en forma de:

```
atributo(concepto) = valor
```

En casos sencillos se utiliza como alternativa:

```
atributo = valor
```

Los valores posibles pueden ser un valor numérico, un valor cualitativo o un conjunto de los anteriores. Seguidamente se describen las características y casos particulares de dicha representación mediante ejemplos.

Dominio de valores excluyentes

Los valores de un atributo normalmente son excluyentes. Se asume que en un determinado momento sólo uno se cumple. Por ejemplo, el dominio de posibles valores del siguiente atributo son excluyentes:

```
fiebre(paciente) = {nula, baja, alta}
```

Dominio multivaluado de valores

En ciertas situaciones un atributo puede tomar varios valores simultáneamente, aunque es preferible reducir estos casos al mínimo para mejorar la eficiencia de los procesos de búsqueda. Por ejemplo, el siguiente atributo es multivaluado:

```
síntoma(paciente) = {leucopenia, disnea, fiebre, neuralgia}
```

Clases de conceptos

Los conceptos se pueden organizar en clases de forma que, mediante herencia, los descendientes de clases compartan propiedades de los valores. Para establecer

relaciones de conceptos con sus clases se suele utilizar un atributo especial (por ejemplo el atributo *clase*). Los valores posibles del atributo *clase* son conceptos. Por ejemplo:

```
clase(hepatitis) = enfermedad-hígado
```

Hechos temporales

Una forma de manejar atributos con valores en el tiempo es asumiendo un segundo argumento del atributo para indicar una referencia temporal (un número, intervalo o nombre):

```
fiebre(paciente,día-1) = normal
fiebre(paciente,día-2) = alta
fiebre(paciente,día-3) = baja
```

Hechos imprecisos o hechos inciertos

Para asociar una medida de imprecisión o de incertidumbre a un hecho se puede manejar valores numéricos como los siguientes: entre 0 y 1 (medida de posibilidad de lógica difusa), entre –1 y 1 (factor de certeza), etc. Dicha medida se asocia al valor:

```
fiebre(paciente) = {normal[0.0], baja[0.3], alta[0.7]}
enfermedad(paciente) = {hepatitis[0.8], colestasis[-1.0]}
```

A.2. Representación de reglas

La sintaxis básica de representación en reglas es la siguiente:

```
atributo(concepto) = valor y
....
atributo(concepto) = valor
→ atributo(concepto) = valor
```

No obstante se consideran variantes más complejas que incluyen variables, operadores de comparación, etc. Seguidamente se indican cada uno de los tipos de reglas considerados mostrando ejemplos.

Regla con operadores de comparación

Incluye comparaciones diferentes de la igualdad en el antecedente de la regla, como mayor, menor, diferente, pertenencia a conjunto, pertenencia a intervalo, etc. Por ejemplo:

```
temperatura(paciente) > 39 → fiebre(paciente) = alta
```

Reglas con expresiones lógicas en antecedente y consecuente

Una regla con negación en consecuente indica la negación de una conclusión y por tanto descarta su valor del conjunto de valores posibles. Por ejemplo:

```
nivel(transaminasas) = normales
→ NO(enfermedad(paciente) = hepatitis)
```

Una regla con disyunción en antecedente normalmente es equivalente a tantas reglas sobre el consecuente como disyunciones haya. Un ejemplo de este tipo de regla es el siguiente:

```
nivel(GPT) = alto o nivel(GOT) = alto
→ nivel(transaminasa) = alto
```

Una regla con disyunción en el consecuente es menos habitual aunque puede ser útil, por ejemplo, para describir una jerarquía. Por ejemplo:

```
enfermedad(paciente) = hígado
→ enfermedad(paciente) = hepatitis o
   enfermedad(paciente) = cirrosis
```

Una regla con conjunción en el consecuente es equivalente a tantas reglas como partes de la conjunción con el mismo antecedente. Por ejemplo:

```
nivel(transaminasas) = altas y
síntoma(paciente) = ictericia
→ enfermedad(paciente) = hepatitis y
   terapia-inmediata(paciente) = aumentar-líquidos
```

Regla con variables

Los valores de atributos se pueden referenciar con variables (con letras en mayúscula X, Y, Z, ...) lo cual aporta generalidad a las reglas. Además pueden incluirse expresiones de comparación en antecedente con dichas variables y expresiones con operadores (aritméticos o funciones) sobre dichas variables. Por ejemplo:

```
ancho(plataforma) = X y largo(plataforma) = Y y X > 20
→ peso(cabina) = 130*(X + Y)/12
```

En este ejemplo en el consecuente se tiene una expresión aritmética. De forma general también pueden presentarse funciones de cualquier tipo (trigonométricas, estadísticas, de acceso a base de datos, de acceso a tabla, etc.).

También pueden considerarse reglas con variables en conceptos de atributos. Es una regla en donde el concepto de un atributo está referenciado de forma general con una variable. Dicha regla representa en una sola un conjunto de reglas, es decir, se entiende que se aplica a todos los descendientes (subclases e instancias) de la clase. Por ejemplo:

```
clase(X) = embalse y sensor-desague(X) = Z y caudal(Z) > 50
→ estado(X) = abierto
```

Regla con consecuente complejo

Se trata de reglas que utilizan predicados sobre hechos para expresar preferencias o acciones. Por ejemplo:

```
localización(paciente)=tropical
→ PREFERIR(enfermedad(paciente)=malaria,
           enfermedad(paciente)=gripe)

retrasado = X y reserva = sí y
refuerzo = Y y zona = final
→ refuerzo-desde-inicio(X,Y)
```

Metarregla

Es una regla que en sus hechos incluye otros elementos de representación (reglas, restricciones, etc.) directamente o por nombre. Por ejemplo una regla que en el consecuente tiene una restricción:

```
modelo(polea-motor) = K3140
→ restricción-relación-ángulo-tracción-motor:
   tracción(motor) < 0.007888 * ángulo(motor) + 0.675
```

Otro ejemplo de metarregla en donde se utiliza el nombre de una restricción en el antecedente (además de un consecuente complejo):

```
VIOLACIÓN(restricción-máxima-altura-pila-contrapeso)
→ INCREMENTAR-HASTA(altura(marco),6, máxima-altura-marco)
```

A.3. Representación de restricciones

Las restricciones establecen relaciones de equilibrio entre atributos. Para ello utilizan comparadores (igual, menor, mayor, etc.) y funciones (operaciones aritméticas, funciones trigonométricas, funciones estadística, etc.).

Los atributos pueden ser numéricos o cualitativos. Las funciones que operan sobre valores cualitativos deben ser formuladas de forma explícita en forma de tabla o conjunto de reglas.

Las restricciones se pueden procesar de dos formas alternativas: (1) *verificación simple*, es decir, comprobación de si se satisfacen las relaciones expresadas teniendo en cuenta que se conocen los valores de todos los atributos, (2) *deducción*

por propagación de valores, es decir, asumiendo que se conocen los valores de parte de los atributos, las restricciones se utilizan para deducir los valores del resto de atributos. La segunda opción de forma general es un problema complejo que, para que pueda ser tratable computacionalmente, requiere realizar ciertas simplificaciones.

Restricción de límite superior o inferior

Son restricciones habituales en sistemas de diseño paramétrico. En ocasiones, todas las restricciones de un modelo se pueden reducir a éstas (haciendo uso además de representación en reglas). Establecen un valor superior o inferior de un determinado atributo. Por ejemplo:

```
altura(cabina) =< 240
```

Restricción entre varios atributos

Incluyen más de un atributo y operaciones de composición entre ellos. Por ejemplo:

```
tracción(motor) < 0.007888 * ángulo(motor) + 0.675
```

Restricción condicionada

Es una restricción que se debe verificar sólo si se cumplen determinadas condiciones (puede verse también como una metarregla). Por ejemplo:

```
modelo(polea-motor) = K3140
→ restricción-relación-ángulo-tracción-motor:
   tracción(motor) < 0.007888 * ángulo(motor) + 0.675
```

Restricción numérica de dirección múltiple

Permite obtener el valor de un atributo conocido el de los otros, en cualquier dirección. Por ejemplo:

```
X + Y = Z
```

si es de dirección múltiple, permite conocer Z dado X e Y pero también Y dados X y Z o X dados Y y Z. Para ello, debe haber una representación asociada a los operadores que permita la operación inversa.

Restricción cualitativa de dirección múltiple

Una restricción cualitativa de dirección múltiple puede restringir el dominio de valores de unos atributos a partir de los dominios de otros atributos, en cualquier combinación. Por ejemplo, en el caso de que X, Y, Z, T sean cualitativos con un dominio como `{negativo, cero, positivo}` la restricción:

```
X + Y = Z + T
```

reduce los valores posibles de unas variables a partir de los valores de otras. Para ello la operación suma debe definirse de forma cualitativa en una tabla en donde se indica por ejemplo que `cero + cero` es `cero`, `positivo + positivo` es `positivo`, `positivo + negativo` es `{negativo, cero, positivo}`, etc.

ANEXO B:
Sistemas de Mantenimiento de la Verdad

En este anexo se incluye un resumen del concepto de Sistema de Mantenimiento de la Verdad. Se trata de un tipo de sistemas que pueden ser útiles como apoyo a procedimientos de inferencia. En particular, se mencionan sistemas de tipo JTMS utilizados en diseño y se describen con más detalle los sistemas de tipo ATMS a los que se hace referencia desde el capítulo relacionado con métodos de diagnóstico.

B.1. Introducción

Ciertos sistemas inteligentes simulan un tipo de razonamiento humano basado en el manejo de hipótesis. Estos métodos resuelven problemas considerando algunos hechos en forma de hipótesis o supuestos a partir de los que se deducen conclusiones. Después, los supuestos pueden ser reconsiderados o retraídos para considerar otras alternativas, de forma que es necesario eliminar también todos los hechos deducidos bajo dichos supuestos. Un sistema de mantenimiento de la verdad TMS (Truth Maintenance System) [Forbus, De Kleer, 93] permite realizar de una

forma eficiente este proceso de reconsideración de hipótesis y propagación de hechos deducidos.

Un TMS puede verse como una memoria dinámica asociada a un motor de inferencia. La idea básica es que cada vez que un motor de inferencia llega a una conclusión, se almacena en dicha memoria la justificación de dicha conclusión en forma de implicación. Por ejemplo, si se ha deducido C a partir de A y B, se almacena A, B → C. Si después deduce E a partir de C y D, se almacena C, D → E. Si se deduce Q de M y N se almacena M, N → Q. Supóngase que en ese momento el motor de inferencia desea retraer el hecho A, es decir, reconsiderar el hecho A asumiendo que ahora no es verdadero. En ese caso, apoyándose en las justificaciones, puede concluirse que también deben retraerse los hechos C y E, pero se mantienen otros como el hecho Q. Este proceso de verificación sobre qué hechos se mantienen y qué hechos dejan de deducirse cuando se reconsideran supuestos es la tarea principal que realiza un TMS apoyándose en la memoria de justificaciones suministrada por el motor de inferencia.

En general, los TMS son útiles en las siguiente situaciones:

- *Generar explicaciones*. Permite servir de almacenamiento del encadenamiento lógico seguido por el motor de inferencia y a partir de ello generar explicaciones.

- *Explicar incoherencias*. Un conjunto de hechos puede resultar incoherente entre sí (por ejemplo, el conjunto de deseos de un cliente sobre un vehículo sobre precio, potencia, etc.). Si dichos hechos se analizan mediante las explicaciones que genera un TMS se pueden identificar mejor las razones de dichas incoherencias. Por ejemplo entre un conjunto de hechos {A,B,C,D,E,F,G,H} un TMS puede indicar que la incompatibilidad es debida sólo a {C,F,H}.

- *Reutilización de resultados previos*. Cuando se realiza una búsqueda tentativa, cada vez que se toma una opción correspondiente a una rama del árbol de búsqueda se asume un supuesto que luego puede retraerse. En un proceso de backtracking cronológico, cuando se abandonan ramas de búsqueda que fallan, se desecha información que en algunos casos podría aprovecharse en otras ramas tales como: (1) conclusiones derivadas de procesos costosos de cálculo o (2) presencia de contradicciones entre elecciones. Por ejemplo, si en una rama de búsqueda se tiene la secuencia de supuestos {A, B, C} y, después, se hace marcha atrás a una rama con {A, D} todo lo que se dedujo de la presencia simultánea de {B, C} se pierde por lo que si continúa la búsqueda con la rama {A, D, B, C} es necesario repetir de nuevo las deducciones que se hicieron de {B, C}.

- *Guiar backtracking*. Cuando en un proceso de búsqueda se alcanza una rama de fallo, el análisis de la contradicción mediante un TMS permite identificar cuál es la incoherencia. Por ejemplo, en una rama de búsqueda se han realizado los supuestos según el orden que se presenta a continuación {A,B,C,D,E,F} y se debe retroceder. Un backtracking cronológico reconsideraría F (el último). Sin embargo, si el TMS indica que D y E son inconsistentes entonces el backtracking puede hacerse a una alternativa de E. Esto se conoce como estrategia de backtracking dirigida por dependencia.

- *Razonamiento por defecto*. Un TMS permite asumir información por defecto cuando se dispone de información insuficiente y reconsiderar adecuadamente las conclusiones cuando se agrega nueva información realizando un tipo de razonamiento no monótono.

B.2. Terminología

Un TMS habitualmente utiliza los siguientes términos:

- *Nodo*: es un elemento básico de la estructura del TMS. Puede ser un hecho, una regla, una restricción, etc. es decir un elemento utilizado por el motor de inferencia. Los nodos pueden ser: (1) *premisa*, es un nodo que siempre es cierto (no puede retraerse), (2) *contradicción*: es un nodo que identifica una contradicción, (3) *supuesto*: es un nodo que por el momento se considera que es cierto pero que después puede ser retraído.

- *Justificación*: es una implicación del tipo P1, ..., Pn → Q en donde Pi es un nodo antecedente y Q el nodo consecuente.

- *Etiqueta*: recoge información sobre la creencia que se tiene en este momento en ese nodo. Puede ser diferente en cada tipo de TMS (JTMS, ATMS, etc.)

B.3. Operación en un TMS

Normalmente un TMS opera de forma conjunta con un motor de inferencia. El esquema de comunicación entre ambos es que el motor de inferencia suministra periódicamente justificaciones y supuestos al TMS. El TMS mantiene y actualiza convenientemente una base de datos que recoge las justificaciones y las etiquetas.

Como resultado, el TMS indica al motor de inferencia la creencias en ciertos nodos (dadas por las etiquetas) y la presencia de contradicciones. El TMS puede generar como resultado explicaciones sobre cómo se alcanzó un hecho. También puede generar los supuestos que explican una contradicción (lo cual es útil para dirigir un *backtracking*).

B.4. ATMS (Assumption-based TMS)

Un ATMS (Assumption-based TMS) [De Kleer, 86] es un TMS que mantiene una estructura de datos eficiente para manejar diferentes conjuntos de supuestos. Esto es especialmente útil cuando se desea que el motor de inferencia haga frecuentemente cambios en los supuestos (como sucede por ejemplo en diagnóstico) y no se desea que cada cambio implique recalcular otra vez de nuevo las etiquetas como ocurre en los TMS de tipo JTMS. Si dicha frecuencia de cambios en supuestos es alta comparada con las veces que se desea conocer la credibilidad en un nodo entonces es preferible utilizar el enfoque ATMS ya que evita desperdiciar mucho tiempo recalculando etiquetas que el motor de inferencia no va a necesitar conocer.

Un ATMS almacena una *etiqueta* para cada nodo más compleja que otros TMS que incluye conjuntos de supuestos a partir de los cuales el nodo puede deducirse. Tiene el siguiente formato:

<*N*, {{*A1*, ..., *An*}, ..., {*B1*, ..., *Bm*}}>

en donde *N* es el identificador del nodo. En la etiqueta se incluye un conjunto de entornos. Cada *entorno* es un conjunto de supuestos consistentes con el nodo *N*. Ejemplos de etiquetas:

- nodo intermedio: `<P, {{S1, S2}, {S3, S4}}>`
- nodo premisa: `<q, {{}}>`
- nodo no deducible (inalcanzable): `<n, {}>`
- supuestos incompatibles: `<no-good, {{S2, S3}, {S5, S7}}>`

B.5. Algoritmo de un ATMS

Para realizar el proceso de actualización de etiquetas que se lleva a cabo con el ATMS se puede aplicar el siguiente algoritmo. El dato de entrada es una justificación J de la forma P1, ..., Pn → Q (en dicho antecedente algunos elementos son supuestos):

1. Obtener el conjunto E de Q de la siguiente forma:
   a. Se incluyen en E todos los conjuntos que son unión de un entorno de la etiqueta del nodo P1, un entorno de la etiqueta del nodo P2, ... y un entorno de la etiqueta del nodo Pn, haciendo todas las combinaciones posibles.
   b. Se eliminan de E (1) superconjuntos de algún conjunto de E (2) superconjuntos de algún entornos de la etiqueta de un no-good.
   c. Se actualiza la etiqueta. Para ello, si A es la etiqueta anterior de Q: (1) se eliminan de E los entornos repetidos en A o superconjuntos de entornos de A, (2) se eliminan de A los entornos superconjuntos de E, (3) la nueva etiqueta es la unión de E y de A.
2. Si E es igual que la etiqueta anterior, finalizar.
3. Si Q es una contradicción marcar todos los entornos de E como no-good y recorrer todos los nodos para eliminar de sus etiquetas los entornos que incluyen algún entorno de E. Finalizar.
4. Si Q no es una contradicción, actualizar recursivamente todas las etiquetas de los nodos que son consecuencia de Q de acuerdo con las justificaciones.

Este algoritmo es adecuado para realizarlo de forma manual en problemas sencillos y para comprender el funcionamiento de un ATMS. No obstante, para su programación es más eficiente hacer una versión incremental tal como se propone en [Forbus, de Kleer, 93]. En el libro [Cuena, 98] se incluyen ejemplos de actualización de etiquetas mediante el procedimiento manual..

B.6. Ejemplo de operación en problema de diagnóstico

Se desarrolla aquí paso a paso el ejemplo del caso de diagnóstico con GDE (ejemplo correspondiente al caso de la figura 4.4 del capítulo de diagnóstico del presente libro). Las entradas que recibe el ATMS son las medidas que se tienen en cada momento junto con las conclusiones a las que llega el proceso de satisfacción de restricciones por propagación de dichas medidas en las restricciones que representan el comportamiento del circuito.

```
1) ESTADO INICIAL

JUSTIFICACIONES:
A=3, C=2, M1 → X=6
B=2, D=3, M2 → Y=6
C=2, E=3, M2 → Z=6
X=6, Y=6, A1 → F=12
Y=6, Z=6, A2 → G=12
ETIQUETAS:
<A=3,{{}}>
<B=2,{{}}>
<C=2,{{}}>
<D=3,{{}}>
<E=3,{{}}>
<X=6,{{M1}}>
<Y=6,{{M2}}>
<Z=6,{{M3}}>
<F=12,{{A1,M1,M2}}>
<G=12,{{A2,M2,M3}}>

2) ENTRADA: F=10

F=12 es una contradicción, por lo que sus entornos se asocian a la etiqueta de no-
good.
```

JUSTIFICACIONES:

A=3, C=2, M1 → X=6

B=2, D=3, M2 → Y=6

C=2, E=3, M2 → Z=6

Y=6, Z=6, A2 → G=12

ETIQUETAS:

<A=3,{{}}>

<B=2,{{}}>

<C=2,{{}}>

<D=3,{{}}>

<E=3,{{}}>

<X=6,{{M1}}>

<Y=6,{{M2}}>

<Z=6,{{M3}}>

<F=10,{{}}>

<G=12,{{A2,M2,M3}}>

<no-good,{{A1,M1,M2}}>

3) ENTRADA: F=10, X=6, A1 → Y=4

JUSTIFICACIONES:

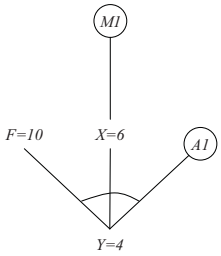
A=3, C=2, M1 → X=6

B=2, D=3, M2 → Y=6

C=2, E=3, M2 → Z=6

Y=6, Z=6, A2 → G=12

F=10,X=6, A1 → Y=4



ETIQUETAS:

<A=3,{{}}>

<B=2,{{}}>

<C=2,{{}}>

<D=3,{{}}>

<E=3,{{}}>

<X=6,{{M1}}>

<Y=4,{{A1,M1}}>

<Y=6,{{M2}}>

<Z=6,{{M3}}>

<F=10,{{}}>

<G=12,{{A2,M2,M3}}>

<no-good,{{A1,M1,M2}}>

4) ENTRADA: F=10, Y=6, A1 → X=4

JUSTIFICACIONES:

A=3, C=2, M1 → X=6

B=2, D=3, M2 → Y=6

C=2, E=3, M3 → Z=6

Y=6, Z=6, A2 → G=12

F=10,X=6, A1 → Y=4

F=10,Y=6, A1 → X=4

ETIQUETAS:

<A=3,{{}}>

<B=2,{{}}>

<C=2,{{}}>

<D=3,{{}}>

<E=3,{{}}>

<X=4,{{A1,M2}}>

<X=6,{{M1}}>

<Y=4,{{A1,M1}}>

<Y=6,{{M2}}>

<Z=6,{{M3}}>

<F=10,{{}}>

<G=12,{{A2,M2,M3}}>

<no-good,{{A1,M1,M2}}>

5) ENTRADA: Y=4, Z=6, A2 → G=10

JUSTIFICACIONES:

A=3, C=2, M1 → X=6

B=2, D=3, M2 → Y=6

C=2, E=3, M3 → Z=6

Y=6, Z=6, A2 → G=12

F=10,X=6, A1 → Y=4

F=10,Y=6, A1 → X=4

Y=4, Z=6, A2 → G=10

ETIQUETAS:

<A=3,{{}}>

<B=2,{{}}>

<C=2,{{}}>

<D=3,{{}}>

<E=3,{{}}>

<X=4,{{A1,M2}}>

<X=6,{{M1}}>

<Y=4,{{A1,M1}}>

<Y=6,{{M2}}>

<Z=6,{{M3}}>

<F=10,{{}}>

<G=10,{{A1,A2,M1,M3}}>

<G=12,{{A2,M2,M3}}>

<no-good,{{A1,M1,M2}}>

6) ENTRADA: G=12

Contradicción con G=10

JUSTIFICACIONES:

A=3, C=2, M1 → X=6

B=2, D=3, M2 → Y=6

C=2, E=3, M3 → Z=6

F=10,X=6, A1 → Y=4

F=10,Y=6, A1 → X=4

ETIQUETAS:

<A=3,{{}}>

<B=2,{{}}>

<C=2,{{}}>

<D=3,{{}}>

<E=3,{{}}>

<X=4,{{A1,M2}}>

<X=6,{{M1}}>

<Y=4,{{A1,M1}}>

<Y=6,{{M2}}>

<Z=6,{{M3}}>

<F=10,{{}}>

<G=12,{{}}>

<no-good,{{A1,M1,M2},{A1,A2,M1,M3}}>

7) ENTRADA: G=12, Y=4, A2 → Z=8

JUSTIFICACIONES:

A=3, C=2, M1 → X=6

B=2, D=3, M2 → Y=6

C=2, E=3, M3 → Z=6

F=10,X=6, A1 → Y=4

F=10,Y=6, A1 → X=4

G=12,Y=4, A2 → Z=8

ETIQUETAS:

<A=3,{{}}>

<B=2,{{}}>

<C=2,{{}}>

<D=3,{{}}>

<E=3,{{}}>

<X=4,{{A1,M2}}>

<X=6,{{M1}}>

<Y=4,{{A1,M1}}>

<Y=6,{{M2}}>

<Z=6,{{M3}}>

<Z=8,{{A1,A2,M1}}>

<F=10,{{}}>

<G=12,{{}}>

<no-good,{{A1,M1,M2},{A1,A2,M1,M3}}>

8) ENTRADA: G=12, Y=6, A2 → Z=6

JUSTIFICACIONES:

A=3, C=2, M1 → X=6

B=2, D=3, M2 → Y=6

C=2, E=3, M3 → Z=6

F=10,X=6, A1 → Y=4

F=10,Y=6, A1 → X=4

G=12,Y=4, A2 → Z=8

G=12,Y=6, A2 → Z=6

ETIQUETAS:

<A=3,{{}}>

<B=2,{{}}>

<C=2,{{}}>

<D=3,{{}}>

<E=3,{{}}>

<X=4,{{A1,M2}}>

<X=6,{{M1}}>

<Y=4,{{A1,M1}}>

<Y=6,{{M2}}>

<Z=6,{{M3},{A2,M2}}>

<Z=8,{{A1,A2,M1}}>

<F=10,{{}}>

<G=12,{{}}>

<no-good,{{A1,M1,M2},{A1,A2,M1,M3}}>

9) ENTRADA: G=12, Z=6, A2 → Y=6

JUSTIFICACIONES:

A=3, C=2, M1 → X=6

B=2, D=3, M2 → Y=6

C=2, E=3, M3 → Z=6

F=10,X=6, A1 → Y=4

F=10,Y=6, A1 → X=4

G=12,Y=4, A2 → Z=8

G=12,Y=6, A2 → Z=6

G=12,Z=6, A2 → Y=6

Aquí se produce propagación con Y=6 hacia X=4 (y hacia Z=6 pero éste sin efecto).

ETIQUETAS:

<A=3,{{}}>

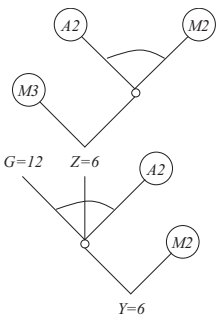
<B=2,{{}}>

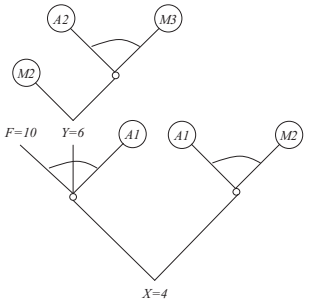
<C=2,{{}}>

<D=3,{{}}>

<E=3,{{}}>

<X=4,{{A1,M2},{A1,A2,M3}}>





<X=6,{{M1}>

<Y=4,{{A1,M1}}>

<Y=6,{{M2},{A2,M3}}>

<Z=6,{{M3},{A2,M2}}>

<Z=8,{{A1,A2,M1}}>

<F=10,{{}}>

<G=12,{{}}>

<no-good,{{A1,M1,M2},{A1,A2,M1,M3}}>

10) ENTRADA: X=6

JUSTIFICACIONES:

B=2, D=3, M2 → Y=6

C=2, E=3, M3 → Z=6

F=10,X=6, A1 → Y=4

G=12,Y=4, A2 → Z=8

G=12,Y=6, A2 → Z=6

G=12,Z=6, A2 → Y=6

ETIQUETAS:

<A=3,{{}}>

<B=2,{{}}>

<C=2,{{}}>

<D=3,{{}}>

<E=3,{{}}>

<X=6,{{}}>

<Y=4,{{A1,M1}}>

<Y=6,{{M2},{A2,M3}}>

<Z=6,{{M3},{A2,M2}}>

<Z=8,{{A1,A2,M1}}>

<F=10,{{}}>

<G=12,{{}}>

<no-good,{{A1,M2},{A1,A2,M3}}>

11) ENTRADA: F=10, X=6 → Y=4

Esta justificación ya está presente. No se producen cambios

JUSTIFICACIONES:

B=2, D=3, M2 → Y=6

C=2, E=3, M3 → Z=6

F=10,X=6, A1 → Y=4

G=12,Y=4, A2 → Z=8

G=12,Y=6, A2 → Z=6

G=12,Z=6, A2 → Y=6

ETIQUETAS:

<A=3,{{}}>

<B=2,{{}}>

<C=2,{{}}>

<D=3,{{}}>

<E=3,{{}}>

<X=6,{{}}>

<Y=4,{{A1,M1}}>

<Y=6,{{M2},{A2,M3}}>

<Z=6,{{M3},{A2,M2}}>

<Z=8,{{A1,A2,M1}}>

<F=10,{{}}>

<G=12,{{}}>

<no-good,{{A1,M2},{A1,A2,M3}}>

ANEXO C:
Test de autoevaluación

Se incluye en este anexo un conjunto de 40 preguntas en forma de test para autoevaluación. Cada pregunta tiene tres respuestas posibles, siendo sólo una de ellas válida. Al final del anexo se presentan las soluciones.

1. En el problema para determinar si se acepta o se rechaza conceder un crédito bancario a un cliente se puede aplicar el método de clasificación heurística con la siguiente estrategia:

(a) estrategia dirigida por objetivos, porque el espacio de soluciones es reducido
(b) estrategia dirigida por los datos, porque no se dispone de toda la información del cliente a priori
(c) estrategia establecer y refinar, dado que se busca establecer o rechazar la hipótesis solución

2. Se sabe que la causa C1 manifiesta los síntomas S1 y S2, y la causa C2 manifiesta los síntomas S2 y S3. Inicialmente se observa el síntoma S2. ¿Qué debe suceder para rechazar la causa C2 durante el proceso de diferenciación del método cubrir y diferenciar?

(a) que no se observe el síntoma S3

(b) que se observe el síntoma S1
(c) que se observe el síntoma S1 y no se observe el S3

3. La base de conocimiento de relaciones de asociación en el método de clasificación heurística:
(a) contiene normalmente criterios subjetivos y/o empíricos para relacionar observaciones con soluciones
(b) contiene normalmente asociaciones entre las soluciones del problema
(c) contiene normalmente relaciones de cálculo para obtener los valores de unos parámetros a partir de otros

4. Considérese una base de conocimiento sobre cálculo de parámetros, de acuerdo con el método proponer y revisar, que contiene las siguientes relaciones: con A y B se calcula C, con D y E se obtiene F, con C y G se obtiene H, con G y F se obtiene I. La forma de procesar dichas relaciones en el método proponer y revisar es:
(a) a partir de los datos A, B, D, E, G se obtienen propuestas de H e I que luego deben ser revisadas
(b) a partir de los objetivos H e I, se interroga por A, B, D, E y G para revisar si satisface las restricciones
(c) las relaciones se consideran como restricciones para evitar bucles y se procesan sin orden prefijado durante la revisión

5. En las bases de conocimiento del método de cubrir y diferenciar, se cumplirá que:
(a) para cada relación causa-manifiesta-efecto existirá su correspondiente relación inversa efecto-evoca-causa
(b) para cada relación efecto-evoca-causa existirá su correspondiente relación inversa causa-manifiesta-efecto
(c) no existe ninguna correspondencia entre ambos tipos de relaciones

6. En un problema de venta de automóviles, en donde se trata de encontrar el tipo de vehículo que mejor encaja con las preferencias de un cliente:

(a) es mejor aplicar clasificación jerárquica porque las preferencias del cliente se pueden organizar en una jerarquía
(b) es mejor aplicar cubrir y diferenciar porque permite encadenar directamente preferencias y vehículos
(c) es mejor aplicar clasificación jerárquica porque los tipos de vehículos se pueden organizar en una jerarquía y, además, permite preguntar las preferencias del cliente de forma progresiva

7. El método de cubrir y diferenciar es más adecuado que el método de clasificación heurística cuando:

(a) se dispone de una estructura causa-efecto que permite relacionar síntomas y causas a diferentes niveles
(b) no se dispone de juicios heurísticos, empíricos o subjetivos para relacionar soluciones con observaciones
(c) el espacio de observaciones, que contiene los tipos de datos, no se puede organizar en una jerarquía

8. Una regla del tipo SI A ENTONCES B, en donde A es una causa y B es un síntoma, es una regla que puede aparecer en:

(a) la base de conocimiento de diferenciación del método cubrir y diferenciar
(b) la base de conocimiento de relaciones efectos-causas del método cubrir y diferenciar
(c) la base de conocimiento de relaciones causales del método de clasificación heurística

9. Para un problema de diseño del mecanismo de ventilación en un edificio, en donde se deben encontrar los valores de los parámetros descriptivos del sistema a diseñar, es posible utilizar como método:

(a) el método proponer y revisar, dado que se trata de un problema de configuración
(b) el método cubrir y diferenciar, porque es un problema de asignación de parámetros por cubrimiento progresivo
(c) el método clasificación heurística, porque se trata de un problema de tipo constructivo

10. En un problema en donde se trata de elegir el color de varios objetos aplicando criterios estéticos globales:
(a) puede utilizarse el método de clasificación heurística porque se elige en un conjunto prefijado de colores
(b) se puede utilizar el método de clasificación jerárquica porque los objetos normalmente se pueden organizar en jerarquía de componentes
(c) se puede utilizar el método proponer y revisar porque se trata de un problema de configuración

11. Los problemas de tipo constructivo se distinguen principalmente de los problemas de tipo clasificativo en que:
(a) seleccionan la solución dentro de un conjunto que se ha construido previamente
(b) construyen de forma dinámica la solución durante la ejecución del método
(c) construyen de forma dinámica la base de conocimiento conforme se resuelven los problemas

12. El problema de diagnóstico de fallos en un ordenador puede verse como un problema de tipo constructivo en donde:
(a) los datos son los síntomas observados y las soluciones son los posibles fallos
(b) los datos son los posibles fallos y los resultados son los síntomas (estrategia dirigida por objetivos)
(c) no es un problema de tipo constructivo

13. El conocimiento de abstracción en el método de clasificación heurística:

(a) se utiliza para abstraer los datos del problema

(b) se utiliza para abstraer las soluciones del problema

(c) se utiliza para abstraer los datos y las soluciones del problema

14. En un problema en donde se vaya aplicar el método de clasificación heurística dirigida por los datos:

(a) se tienen que conocer a priori todos los datos del problema

(b) los datos del problema deben estar organizados previamente en una jerarquía de niveles de abstracción

(c) se pueden conocer parcialmente los datos y, durante la búsqueda de la solución, el método solicitará datos adicionales

15. Si se compara el método proponer y revisar con el método de planificación jerárquica se puede afirmar que:

(a) en los problemas clasificativos en donde se puedan utilizar ambos, es preferible utilizar el segundo por ser más robusto

(b) el primero normalmente es más flexible pero puede ser más difícil de poner a punto que el segundo

(c) el primero es para problemas generales de clasificación y el segundo para clasificación de planes abstractos

16. Si los síntomas S1, S2 se cubren con la causa C1, los síntomas S2, S3 y S4 con la causa C2, y la causa C3 cubre a C1 y a S3, ¿cuál es el conjunto de causas que finalmente establece el método cubrir y diferenciar como explicación de los síntomas S1, S2, S3 y S4?:

(a) C1 y C2

(b) C1, C2 y C3

(c) C2 y C3

17. Considérese una base de conocimiento sobre restricciones de diseño que contiene tres restricciones: C1: A<B, C2: C<D, C3: A<D. Para la restricción C1 hay dos remedios posibles M1 y M4. Para la C2 están M2 y M4. Para la C3 se tiene como remedios M3 y M4. Si, en general, se considera como prioridad el orden de subíndice que se presenta (siendo preferible el valor más pequeño), indicar qué modificaciones se proponen en la fase de remediar, según el método proponer y revisar, si la propuesta es: A=1, B=2, C=1, D=1:

(a) M2, M4
(b) M2, M3, M4
(c) M1, M2, M3, M4

18. En un problema para encontrar la secuencia de medios de transporte necesarios para trasladarse de un lugar a otro se elige el método de planificación jerárquica. En ese caso, en el modelo de conocimiento:

(a) puede haber un especialista por cada medio de transporte, organizados en una jerarquía de niveles de generalidad
(b) dado que los resultados son los medios de transporte, los especialistas no pueden ser los medios de transporte
(c) puede haber un especialista por cada tipo de destino, organizados en una jerarquía de niveles de generalidad

19. En un problema en donde se aplique el método de clasificación heurística dirigida por los objetivos:

(a) los datos se piden progresivamente conforme avanza el proceso de resolución
(b) las soluciones se construyen de forma dinámica durante la ejecución del método
(c) los objetivos dirigen el proceso de búsqueda en una jerarquía de soluciones

20. El método de clasificación jerárquica es apropiado para el siguiente grupo de tres problemas:

(a) diagnóstico de enfermedad de riñón, selección del deporte adecuado para una persona, recomendación de tipo de inversión
(b) configuración de equipos informáticos, detección de fallos mecánicos, planificación de terapia médica
(c) asignación de plazas de aparcamiento, decisión de equipo ganador, planificación temporal de cultivos

21. En un problema para determinar la adecuación del perfil de un candidato a un determinado puesto de trabajo es razonable utilizar como método de resolución de problemas:

(a) clasificación heurística, dado que se trata de un problema de clasificación
(b) proponer y revisar, puesto que permite proponer inicialmente al candidato y después revisar su perfil
(c) cualquier método de tipo constructivo.

22. Se sabe que la causa C1 manifiesta los síntomas S1 y S2, y la causa C2 manifiesta los síntomas S2 y S3. Inicialmente se observa el síntoma S2. ¿Qué debe suceder para no rechazar la causa C2 durante el proceso de diferenciación del método cubrir y diferenciar?

(a) que se observe el síntoma S3
(b) que se observe el síntoma S1
(c) que se observe el síntoma S1 y no se observe el S3

23. La base de conocimiento de relaciones de asociación en el método de clasificación heurística:

(a) asocia clases de observaciones con clases de soluciones
(b) asocia observaciones con clases de observaciones
(c) asocia clases de soluciones con soluciones

24. Considérese una base de conocimiento sobre cálculo de parámetros, de acuerdo con el método proponer y revisar, que contiene las siguientes relaciones: “con X y Y se calcula A”, “con Z y U se obtiene B”, “con A y V se obtiene M”, “con V y B se obtiene N”. ¿Es correcto incluir en dicha base la relación: “con M se calcula X”?

(a) sí es correcto

(b) no es correcto, dado que supondría la existencia de un bucle en los cálculos

(c) no es correcto, porque el parámetro M es un nodo terminal

25. La base de conocimiento de relaciones efectos-causas del método cubrir y diferenciar tiene como contenido:

(a) criterios heurísticos con los que dados unos síntomas se hacen hipótesis de las causas

(b) relaciones que permiten establecer con seguridad total las causas de un determinado síntoma

(c) criterios que permiten hacer hipótesis de síntomas a partir de causas

26. En un problema de recomendación del tipo de deporte más adecuado para una persona:

(a) es mejor aplicar clasificación jerárquica, dado que los deportes se pueden organizar en una jerarquía de niveles de generalidad y permite preguntar las características de la persona de forma progresiva

(b) es mejor aplicar cubrir y diferenciar, en donde los síntomas son los deportes y las causas son las características de la persona

(c) es mejor aplicar proponer y revisar, porque se trata de proponer deportes de acuerdo con las necesidades de las personas

27. Si se compara el método de clasificación jerárquica con el método de cubrir y diferenciar:

(a) el primero es más fácil de calibrar y poner a punto dado que la búsqueda se basa en un descenso jerárquico, mientras que el segundo realiza una búsqueda en un grafo en donde la convergencia es más difícil de garantizar
(b) el segundo es más fácil de calibrar y poner a punto dado que realiza una búsqueda lineal, mientras que el primero realiza una búsqueda jerárquica
(c) a priori, ambos métodos tienen igual dificultad de puesta a punto

28. Las reglas "SI medida M < 10 ENTONCES nivel = bajo", "SI medida M >= 10 y M < 20 ENTONCES nivel = medio", "SI medida M >= 20 ENTONCES nivel=alto" de acuerdo con la forma que presentan pueden aparecer en:

(a) la base de conocimiento de soluciones del método de proponer y revisar
(b) la base de conocimiento de abstracción del método de clasificación heurística
(c) la base de conocimiento de estrategias del método de planificación jerárquica HTN.

29. El problema de asignación de plazas de aparcamiento para una flota de vehículos:

(a) es un problema de clasificación, dado que las plazas de aparcamiento están prefijadas de antemano
(b) es un problema constructivo, dado que se trata de establecer dinámicamente relaciones de tipo vehículo-plaza
(c) se puede contemplar indistintamente como un problema de tipo clasificativo o de tipo constructivo.

30. En un problema en donde se trata establecer los cultivos más adecuados para un conjunto de terrenos, teniendo en cuenta criterios globales a todos los terrenos:

(a) debe utilizarse el método de clasificación heurística porque se elige en un conjunto prefijado de cultivos
(b) se debe utilizar el método de clasificación jerárquica porque los cultivos se pueden organizar en jerarquía
(c) se debe utilizar un método constructivo porque se trata de un problema de asignación

31. Los problemas de tipo constructivo, en relación a los problemas de tipo clasificativo:

(a) son siempre de complejidad inferior a los de tipo clasificativo
(b) son siempre de complejidad mayor que los de tipo clasificativo
(c) la complejidad depende de cada caso concreto; habitualmente la complejidad de los problemas constructivos es mayor pero, por ejemplo, algunos problemas de clasificación múltiple pueden ser intratables.

32. Se plantea un problema en donde se dispone de 5 opciones posibles para realizar una posible inversión económica, teniendo en cuenta diversos factores como riesgo asumido, horizonte temporal, etc. En dicho problema se podría aplicar un método:

(a) de tipo clasificativo, en donde el espacio de soluciones es el conjunto de opciones posibles de inversión
(b) de planificación, porque se trata de un problema de planificación de inversiones
(c) constructivo, porque se deben construir las opciones de inversión de forma dinámica.

33. La base de conocimiento de estrategias de combinación en el método de cubrir y diferenciar:

(a) contiene los criterios generales para establecer combinaciones de síntomas
(b) contiene excepciones a los criterios generales para establecer combinaciones de causas que explican los síntomas
(c) contiene restricciones cuya combinación permite proponer nuevos síntomas

34. En un problema en donde se vaya aplicar el método de clasificación jerárquica, al menos:

(a) el espacio de soluciones se debe poder organizar en una jerarquía
(b) el espacio de datos se debe poder organizar en una jerarquía
(c) las asociaciones deben poder organizarse en una jerarquía

35. El método proponer y revisar es adecuado para:

(a) proponer la enfermedad posible de un paciente a partir de unos síntomas observados
(b) elegir un cultivo posible (dentro de un conjunto prefijado) para un determinado terreno
(c) ninguno de los dos anteriores

36. Considerar las siguientes relaciones de tipo causal: los síntomas S1y S2 se cubren con la causa A, los síntomas S2 y S3 con la causa B, los síntomas S4 y S5 con la causa C, la causa D cubre a A y B, la causa E cubre a C y a S6. ¿Cuál es el conjunto de causas que finalmente establece el método cubrir y diferenciar como explicación de los síntomas S1, S2, S3, S4, S5 y S6?:

(a) A, B, C, D y E
(b) D y E
(c) D o E

37. Se desea aplicar el método de cubrir y diferenciar al problema de diagnóstico de averías de un motor de avión a partir de los datos medidos en un banco de pruebas. Para ello, en la base de conocimiento circunstancial:

(a) se puede incluir conocimiento sobre probabilidades a priori de cada tipo de avería, lo que permitirá seleccionar las más frecuentes en primer lugar.
(b) se puede incluir conocimiento de abstracción de los datos de los sensores del banco de pruebas, para recoger la circunstancia de la avería.
(c) se puede incluir relaciones causa-efecto (averías que deben presentar datos del banco de prueba) como complemento a su inversa efecto-causa.

38. El problema de asignación de personas a oficinas en la organización de un edificio de una empresa se puede ver como un problema de tipo clasificativo en donde:

(a) los datos son las personas y las soluciones las oficinas
(b) los datos son las oficinas y las soluciones son las personas
(c) no es un problema de tipo clasificativo

39. En un problema de decisión sobre 4 tipos de estudios universitarios que podría elegir un estudiante, es preferible:

(a) aplicar el método de clasificación heurística dirigido por los datos, aunque no se disponga inicialmente de todos los datos sobre el estudiante
(b) aplicar el método de clasificación heurística dirigido por objetivos, dado que el espacio de soluciones es reducido y se puede preguntar progresivamente los datos del estudiante
(c) aplicar el método de clasificación jerárquica

40. El método de cubrir y diferenciar es más adecuado para:

(a) problemas de diagnóstico en donde se dispone de conocimiento sobre relaciones causa-efecto

(b) problemas de diagnóstico aunque no se disponga de conocimiento sobre relaciones causa-efecto

(c) problemas de diagnóstico y problemas de asignación

La siguiente tabla indica las respuestas válidas a las preguntas del test:

Número de pregunta	**Respuesta correcta**
1	a
2	a
3	a
4	a
5	a
6	c
7	a
8	a
9	a
10	c
11	b
12	c
13	a
14	a
15	b
16	c
17	a
18	a
19	a
20	a
21	a
22	a
23	a
24	b
25	a
26	a
27	a
28	b
29	b
30	c
31	c
32	a
33	b
34	a
35	c
36	b
37	a
38	c
39	b
40	a